石山园

Spark入门实战系列--6.SparkSQL（上）--SparkSQL简介

【注】该系列文章以及使用到安装包/测试数据 可以在《倾情大奉送--Spark入门实战系列》获取

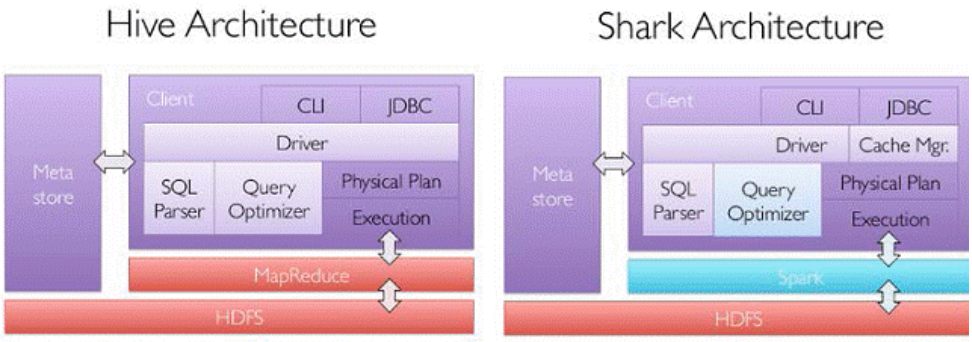
1、SparkSQL的发展历程

1.1 Hive and Shark

SparkSQL的前身是Shark，给熟悉RDBMS但又不理解MapReduce的技术人员提供快速上手的工具，Hive应运而生，它是当时唯一运行在Hadoop上的SQL-on-Hadoop工具。但是MapReduce计算过程中大量的中间磁盘落地过程消耗了大量的I/O，降低的运行效率，为了提高SQL-on-Hadoop的效率，大量的SQL-on-Hadoop工具开始产生，其中表现较为突出的是：

- MapR的Drill
- Cloudera的Impala
- Shark

其中Shark是伯克利实验室Spark生态环境的组件之一，它修改了下图所示的右下角的内存管理、物理计划、执行三个模块，并使之能运行在Spark引擎上，从而使得SQL查询的速度得到10-100倍的提升。



1.2 Shark和SparkSQL

但是，随着Spark的发展，对于野心勃勃的Spark团队来说，Shark对于Hive的太多依赖（如采用Hive的语法解析器、查询优化器等等），制约了Spark的One Stack Rule Them All的既定方针，制约了Spark各个组件的相互集成，所以提出了SparkSQL项目。SparkSQL抛弃原有Shark的代码，汲取了Shark的一些优点，如内存列存储（In-Memory Columnar Storage）、Hive兼容性等，重新开发了SparkSQL代码；由于摆脱了对Hive的依赖性，SparkSQL无论在数据兼容、性能优化、组件扩展方面都得到了极大的方便，真可谓“退一步，海阔天空”。

- **数据兼容方面** 不但兼容Hive，还可以从RDD、parquet文件、JSON文件中获取数据，未来版本甚至支持获取RDBMS数据以及cassandra等NOSQL数据；
- **性能优化方面** 除了采取In-Memory Columnar Storage、byte-code generation等优化技术外、将会引进Cost Model对查询进行动态评估、获取最佳物理计划等等；
- **组件扩展方面** 无论是SQL的语法解析器、分析器还是优化器都可以重新定义，进行扩展。

2014年6月1日Shark项目和SparkSQL项目的主持人Reynold Xin宣布：停止对Shark的开发，团队将所有资源放SparkSQL项目上，至此，Shark的发展画上了句话，但也因此发展出两个直线：SparkSQL和Hive on Spark。

个人简介

郭景瞻，博客以家乡石山园为名，大数据图书作者，著《图解Spark：核心技术与案例实战》。

Spark核心技术与案例
[《图解Spark：核心技术与案例实战》]

昵称：shishanyuan
园龄：8年2个月
荣誉：推荐博客
粉丝：659
关注：16
+加关注

< 2018年3月 >

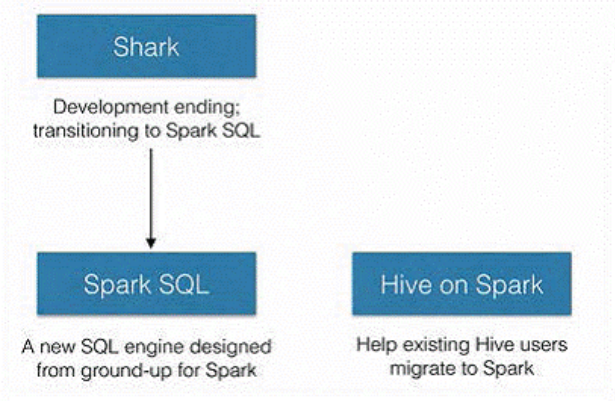
日	一	二	三	四	五	六
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

搜索

找找看

随笔分类(107)

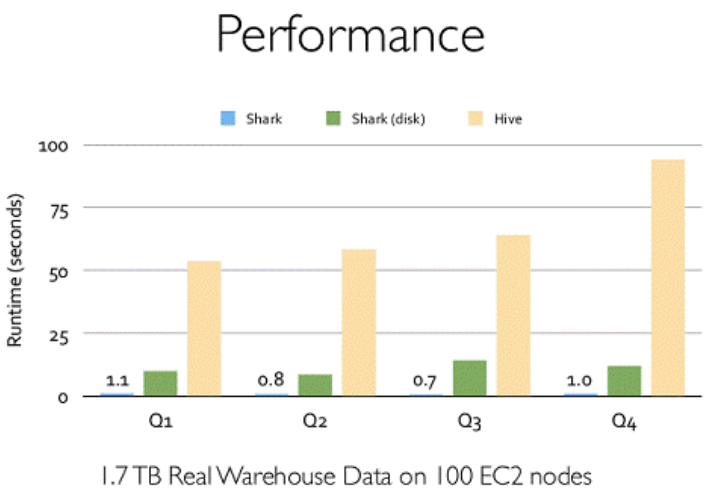
10.《图解Spark:核心技术与案例实战》(10)
20.Spark入门实战系列(19)
27.Spark精彩文章(6)
28.Databricks博客翻译(7)
30.Hadoop入门进阶课程(13)
40.持续集成(8)
50.图解Oracle10g备份恢复系列(20)
51.基于OracleLogminer数据同步(4)
58.数据库优化(1)
90.Hadoop数据分析平台(12)
91.杂项(7)



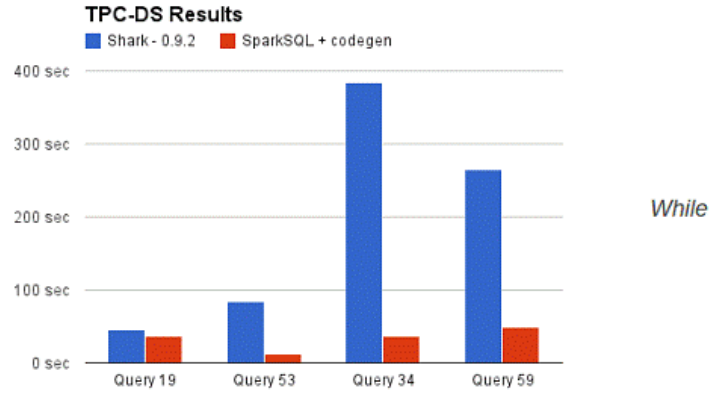
其中SparkSQL作为Spark生态的一员继续发展，而不再受限于Hive，只是兼容Hive；而Hive on Spark是一个Hive的发展计划，该计划将Spark作为Hive的底层引擎之一，也就是说，Hive将不再受限于一个引擎，可以采用Map-Reduce、Tez、Spark等引擎。

1.3 SparkSQL的性能

Shark的出现，使得SQL-on-Hadoop的性能比Hive有了10-100倍的提高：



那么，摆脱了Hive的限制，SparkSQL的性能又有怎么样的表现呢？虽然没有Shark相对于Hive那样瞩目地性能提升，但也表现得非常优异：



为什么SparkSQL的性能会得到怎么大的提升呢？主要SparkSQL在下面几点做了优化：

A：内存列存储（In-Memory Columnar Storage）

SparkSQL的表数据在内存中存储不是采用原生态的JVM对象存储方式，而是采用内存列存储，如下图所示。

5. Spark入门实战系列--8.Spark MLlib（下）--机器学习库SparkMLlib实战(18)

该存储方式无论在空间占用量和读取吞吐率上都占有很大优势。

对于原生态的JVM对象存储方式，每个对象通常要增加12-16字节的额外开销，对于一个270MB的TPC-H lineitem table数据，使用这种方式读入内存，要使用970MB左右的内存空间（通常是2～倍于原生数据空间）；另外，使用这种方式，每个数据记录产生一个JVM对象，如果是大小为200的数据记录，32G的堆栈将产生1.6亿个对象，这么多的对象，对于GC来说，可能要消耗几分钟的时间来处理（JVM的垃圾收集时间与堆栈中的对象数量呈线性相关）。显然这种内存存储方式对于基于内存计算的Spark来说，很昂贵也负担不起。

对于内存列存储来说，将所有原生数据类型的列采用原生数组来存储，将Hive支持的复杂数据类型（如array、map等）先序化后并接成一个字节数组来存储。这样，每个列创建一个JVM对象，从而导致可以快速的GC和紧凑的数据存储；额外的，还可以使用低廉CPU开销的高效压缩方法（如字典编码、行长度编码等压缩方法）降低内存开销；更有趣的是，对于分析查询中频繁使用的聚合特定列，性能会得到很大的提高，原因就是这些列的数据放在一起，更容易读入内存进行计算。

B：字节码生成技术（bytecode generation，即CG）

在数据库查询中有一个昂贵的操作是查询语句中的表达式，主要是由于JVM的内存模型引起的。比如如下一个查询：

SELECT a + b FROM table

在这个查询里，如果采用通用的SQL语法途径去处理，会先生成一个表达式树（有两个节点的Add树，参考后面章节），在物理处理这个表达式树的时候，将会如图所示的7个步骤：

1. 调用虚函数Add.eval()，需要确认Add两边的数据类型
2. 调用虚函数a.eval()，需要确认a的数据类型
3. 确定a的数据类型是Int，装箱
4. 调用虚函数b.eval()，需要确认b的数据类型
5. 确定b的数据类型是Int，装箱
6. 调用Int类型的Add
7. 返回装箱后的计算结果

其中多次涉及到虚函数的调用，虚函数的调用会打断CPU的正常流水线处理，减缓执行。

Spark1.1.0在catalyst模块的expressions增加了codegen模块，如果使用动态字节码生成技术（配置spark.sql.codegen参数），SparkSQL在执行物理计划的时候，对匹配的表达式采用特定的代码，动态编译，然后运行。如上例子，匹配到Add方法：

```
case Add(e1, e2) =>      (e1, e2) evaluate { case (eval1, eval2) => q"$eval1 + $eval2"
```

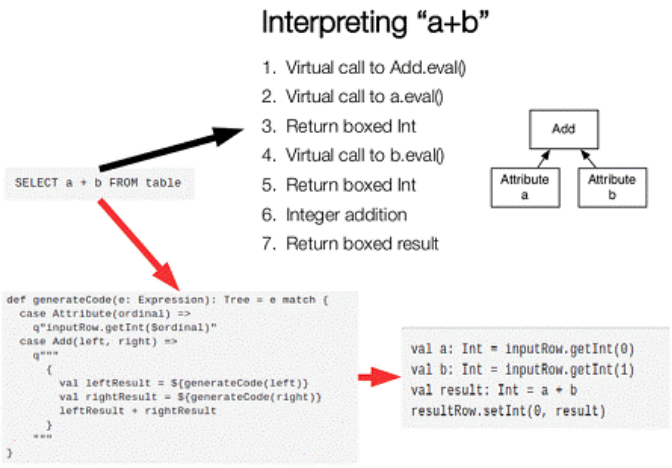
然后，通过调用，最终调用：

```
def evaluateAs(resultType: DataType)(f: (TermName, TermName) => Tree): Seq[Tree] = {
  // TODO: Right now some timestamp tests fail if we enforce this...
  if (expressions._1.dataType != expressions._2.dataType) {
    log.warn(s"${expressions._1.dataType} != ${expressions._2.dataType}")
  }

  val eval1 = expressionEvaluator(expressions._1)
  val eval2 = expressionEvaluator(expressions._2)
  val resultCode = f(eval1.primitiveTerm, eval2.primitiveTerm)

  eval1.code ++ eval2.code ++
  q"""
    val $nullTerm = ${eval1.nullTerm} || ${eval2.nullTerm}
    val $primitiveTerm: ${termForType(resultType)} =
      if($nullTerm) {
        ${defaultPrimitive(resultType)}
      } else {
        $resultCode.asInstanceOf[${termForType(resultType)}]
      }
  """.children : Seq[Tree]
}
```

最终实现效果类似如下伪代码：

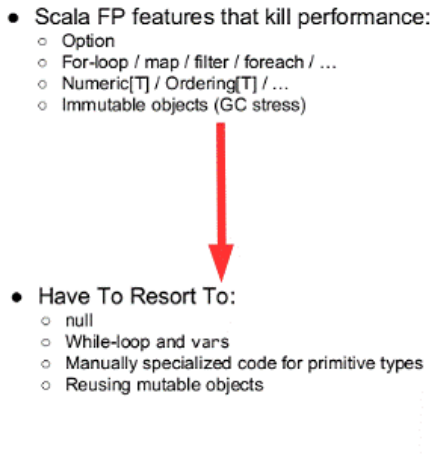
```
val a: Int = inputRow.getInt(0)
val b: Int = inputRow.getInt(1)
val result: Int = a + b
resultRow.setInt(0, result)
```

对于Spark1.1.0，对SQL表达式都作了CG优化，具体可以参看codegen模块。CG优化的实现主要还是依靠scala2.10的运行时放射机制（runtime reflection）。对于SQL查询的CG优化，可以简单地用下图来表示：



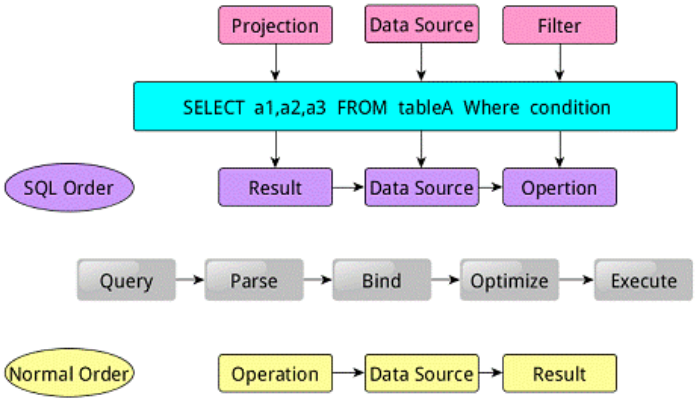
C：Scala代码优化

另外，SparkSQL在使用Scala编写代码的时候，尽量避免低效的、容易GC的代码；尽管增加了编写代码的难度，但对于用户来说，还是使用统一的接口，没受到使用上的困难。下图是一个Scala代码优化的示意图：



2、 SparkSQL运行架构

类似于关系型数据库，SparkSQL也是语句也是由Projection（a1，a2，a3）、Data Source（tableA）、Filter（condition）组成，分别对应sql查询过程中的Result、Data Source、Operation，也就是说SQL语句按Result-->Data Source-->Operation的次序来描述的。



当执行SparkSQL语句的顺序为：

1.对读入的SQL语句进行解析（Parse），分辨出SQL语句中哪些词是关键词（如SELECT、FROM、WHERE），哪些是表达式、哪些是Projection、哪些是Data Source等，从而判断SQL语句是否规范；
2.将SQL语句和数据库的数据字典（列、表、视图等等）进行绑定（Bind），如果相关的Projection、Data Source等都是存在的话，就表示这个SQL语句是可以执行的；
3.一般的数据库会提供几个执行计划，这些计划一般都有运行统计数据，数据库会在这些计划中选择一个最优计划（Optimize）；
4.计划执行（Execute），按Operation-->Data Source-->Result的次序来进行的，在执行过程有时候甚至不需要读取物理表就可以返回结果，比如重新运行刚运行过的SQL语句，可能直接从数据库的缓冲池中获取返回结果。

2.1 Tree和Rule

SparkSQL对SQL语句的处理和关系型数据库对SQL语句的处理采用了类似的方法，首先会将SQL语句进行解析（Parse），然后形成一个Tree，在后续的如绑定、优化等处理过程都是对Tree的操作，而操作的方法是采用Rule，通过模式匹配，对不同类型的节点采用不同的操作。在整个sql语句的处理过程中，Tree和Rule相互配合，完成了解析、绑定（在SparkSQL中称为Analysis）、优化、物理计划等过程，最终生成可以执行的物理计划。

2.1.1 Tree

- Tree的相关代码定义在sql/catalyst/src/main/scala/org/apache/spark/sql/catalyst/trees
- Logical Plans、Expressions、Physical Operators都可以使用Tree表示
- Tree的具体操作是通过TreeNode来实现的
  - SparkSQL定义了catalyst.trees的日志，通过这个日志可以形象的表示出树的结构
  - TreeNode可以使用scala的集合操作方法（如foreach, map, flatMap, collect等）进行操作
  - 有了TreeNode，通过Tree中各个TreeNode之间的关系，可以对Tree进行遍历操作，如使用transformDown、transformUp将Rule应用到给定的树段，然后用结果替代旧的树段；也可以使用transformChildrenDown、transformChildrenUp对一个给定的节点进行操作，通过迭代将Rule应用到该节点以及子节点。
- TreeNode可以细分成三种类型的Node：
  - UnaryNode 一元节点，即只有一个子节点。如Limit、Filter操作
  - BinaryNode 二元节点，即有左右子节点的二叉节点。如Jion、Union操作
  - LeafNode 叶子节点，没有子节点的节点。主要用户命令类操作，如SetCommand

2.1.2 Rule

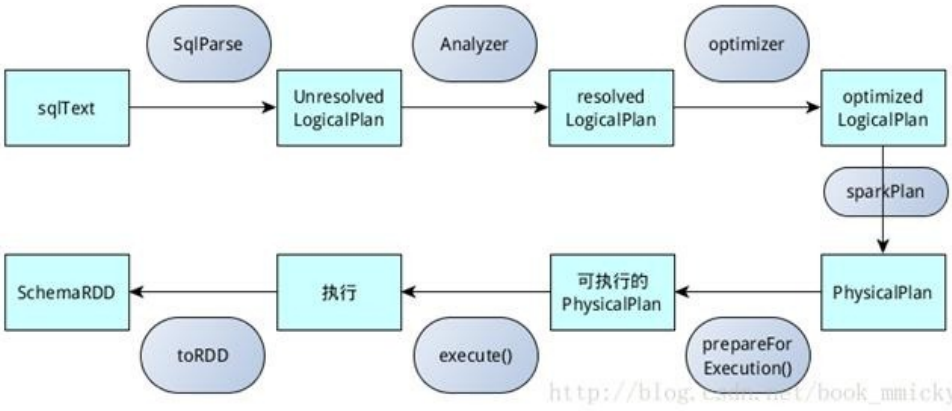
- Rule的相关代码定义在sql/catalyst/src/main/scala/org/apache/spark/sql/catalyst/rules
- Rule在SparkSQL的Analyzer、Optimizer、SparkPlan等各个组件中都有应用到
- Rule是一个抽象类，具体的Rule实现是通过RuleExecutor完成
- Rule通过定义batch和batchs，可以简便的、模块化地对Tree进行transform操作
- Rule通过定义Once和FixedPoint，可以对Tree进行一次操作或多次操作（如对某些Tree进行多次迭代操作的时候，达到FixedPoint次数迭代或达到前后两次的树结构没变化才停止操作，具体参看RuleExecutor.apply）

2.2 sqlContext和hiveContext的运行过程

SparkSQL有两个分支，sqlContext和hiveContext，sqlContext现在只支持SQL语法解析器（SQL-92语法）；hiveContext现在支持SQL语法解析器和hivesql语法解析器，默认为hiveSQL语法解析器，用户可以通过配置切换成SQL语法解析器，来运行hiveSQL不支持的语法，

2.2.1 sqlContext的运行过程

sqlContext总的一个过程如下图所示：
1.SQL语句经过SqlParse解析成UnresolvedLogicalPlan；
2.使用analyzer结合数据数据字典（catalog）进行绑定，生成resolvedLogicalPlan；
3.使用optimizer对resolvedLogicalPlan进行优化，生成optimizedLogicalPlan；
4.使用SparkPlan将LogicalPlan转换成PhysicalPlan；
5.使用prepareForExecution()将PhysicalPlan转换成可执行物理计划；
6.使用execute()执行可执行物理计划；
7.生成SchemaRDD。
在整个运行过程中涉及到多个SparkSQL的组件，如SqlParse、analyzer、optimizer、SparkPlan等等



2.2.2hiveContext的运行过程

hiveContext总的一个过程如下图所示：

1.SQL语句经过HiveQl.parseSql解析成Unresolved LogicalPlan，在这个解析过程中对hiveql语句使用getAst()获取AST树，然后再进行解析；

2.使用analyzer结合数据hive源数据Metastore（新的catalog）进行绑定，生成resolved LogicalPlan；

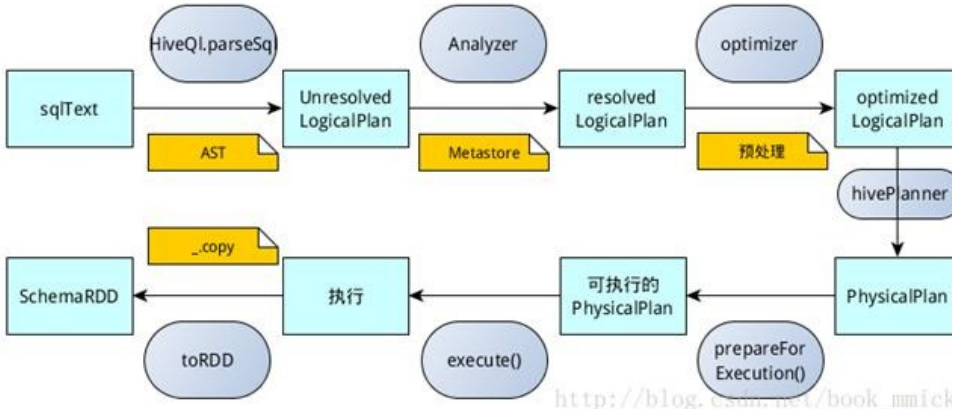
3.使用optimizer对resolved LogicalPlan进行优化，生成optimized LogicalPlan，优化前使用了ExtractPythonUdfs(catalog.PreInsertionCasts(catalog.CreateTables(analyzed)))进行预处理；

4.使用hivePlanner将LogicalPlan转换成PhysicalPlan；

5.使用prepareForExecution()将PhysicalPlan转换成可执行物理计划；

6.使用execute()执行可执行物理计划；

7.执行后，使用map(_.copy)将结果导入SchemaRDD。

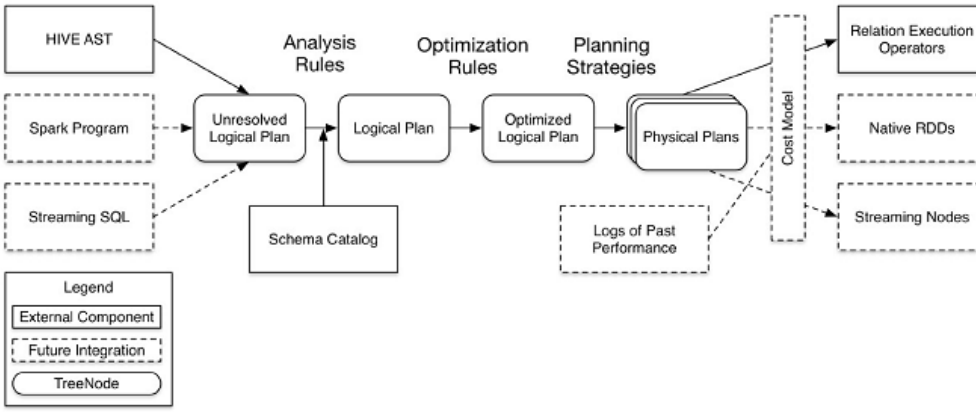


2.3 catalyst优化器

SparkSQL1.1总体上由四个模块组成：core、catalyst、hive、hive-Thriftserver：

- core处理数据的输入输出，从不同的数据源获取数据（RDD、Parquet、json等），将查询结果输出成schemaRDD；
- catalyst处理查询语句的整个处理过程，包括解析、绑定、优化、物理计划等，说其是优化器，还不如说是查询引擎；
- hive对hive数据的处理
- hive-ThriftServer提供CLI和JDBC/ODBC接口

在这四个模块中，catalyst处于最核心的部分，其性能优劣将影响整体的性能。由于发展时间尚短，还有很多不足的地方，但其插件式的设计，为未来的发展留下了很大的空间。下面是catalyst的一个设计图：



其中虚线部分是以后版本要实现的功能，实线部分是已经实现的功能。从上图看，catalyst主要的实现组件有：

●sqlParse，完成sql语句的语法解析功能，目前只提供了一个简单的sql解析器；

●Analyzer，主要完成绑定工作，将不同来源的Unresolved LogicalPlan和数据元数据（如hive metastore、Schema catalog）进行绑定，生成resolved LogicalPlan；

●optimizer对resolved LogicalPlan进行优化，生成optimized LogicalPlan；

● Planner将LogicalPlan转换成PhysicalPlan；

● CostModel，主要根据过去的性能统计数据，选择最佳的物理执行计划

这些组件的基本实现方法：

● 先将sql语句通过解析生成Tree，然后在不同阶段使用不同的Rule应用到Tree上，通过转换完成各个组件的功能。

● Analyzer使用Analysis Rules，配合数据元数据（如hive metastore、Schema catalog），完善Unresolved LogicalPlan的属性而转换成resolved LogicalPlan；

● optimizer使用Optimization Rules，对resolved LogicalPlan进行合并、列裁剪、过滤器下推等优化作业而转换成optimized LogicalPlan；

● Planner使用Planning Strategies，对optimized LogicalPlan

3、SparkSQL CLI

CLI（Command-Line Interface，命令行界面）是指可在用户提示符下键入可执行指令的界面，它通常不支持鼠标，用户通过键盘输入指令，计算机接收到指令后予以执行。Spark CLI指的是使用命令界面直接输入SQL命令，然后发送到Spark集群进行执行，在界面中显示运行过程和最终的结果。

Spark1.1相较于Spark1.0最大的差别就在于Spark1.1增加了Spark SQL CLI和ThriftServer，使得Hive用户还有用惯了命令行的RDBMS数据库管理员较容易地上手，真正意义上进入了SQL时代。

【注】Spark CLI和Spark Thrift Server实验环境为第二课《Spark编译与部署（下）--Spark编译安装》所搭建

3.1 运行环境说明

3.1.1 硬软件环境

- 主机操作系统：Windows 64位，双核4线程，主频2.2G，10G内存
- 虚拟软件：VMware® Workstation 9.0.0 build-812388
- 虚拟机操作系统：CentOS 64位，单核
- 虚拟机运行环境：
  - JDK：1.7.0_55 64位
  - Hadoop：2.2.0（需要编译为64位）
  - Scala：2.11.4
  - Spark：1.1.0（需要编译）
  - Hive：0.13.1

3.1.2 机器网络环境

集群包含三个节点，节点之间可以免密码SSH访问，节点IP地址和主机名分布如下：

序号	IP地址	机器名	类型	核数/内存	用户名	目录
1	192.168.0.61	hadoop1	NN/DN/RM Master/Worker	1核/3G	hadoop	/app 程序所在路径 /app/scala-...

2	192.168.0.62	hadoop2	DN/NM/Worker	1核/2G	hadoop	/app/hadoop /app/complied
3	192.168.0.63	hadoop3	DN/NM/Worker	1核/2G	hadoop	

3.2 配置并启动

3.2.1 创建并配置hive-site.xml

在运行Spark SQL CLI中需要使用到Hive Metastore，故需要在Spark中添加其uris。具体方法是在SPARK_HOME/conf 目 录 下 创 建 hive-site.xml 文 件 ， 然 后 在 该 配 置 文 件 中 ， 添 加hive.metastore.uris属性，具体如下：

```
<configuration>
  <property>
    <name>hive.metastore.uris</name>
    <value>thrift://hadoop1:9083</value>
    <description>Thrift URI for the remote metastore. Used by metastore client to connect
      to remote metastore.</description>
  </property>
</configuration>
```

```
[hadoop@hadoop1 ~]$ cd /app/hadoop/spark-1.1.0/conf
[hadoop@hadoop1 conf]$ ls
fairscheduler.xml.template  log4j.properties.template    spark-defaults.conf.template
hive-site-bak.xml           metrics.properties.template  spark-env.sh
hive-site.xml               slaves                       spark-env.sh.template
[hadoop@hadoop1 conf]$ cat hive-site.xml
<configuration>
  <property>
    <name>hive.metastore.uris</name>
    <value>thrift://hadoop1:9083</value>
    <description>Thrift URI for the remote metastore. Used by metastore client to connect t
o remote metastore.</description>
  </property>
</configuration>
[hadoop@hadoop1 conf]$
```

3.2.2 启动Hive

在使用Spark SQL CLI之前需要启动Hive Metastore（如果数据存放在HDFS文件系统，还需要启动Hadoop的HDFS），使用如下命令可以使Hive Metastore启动后运行在后台，可以通过jobs查询：

```
$nohup hive --service metastore > metastore.log 2>&1 &
```

```
[hadoop@hadoop1 ~]$ $nohup hive --service metastore > metastore.log 2>&1 &
[1] 7014
[hadoop@hadoop1 ~]$ jobs
[1]+  Running                 $nohup hive --service metastore > metastore.log 2>&1 &
[hadoop@hadoop1 ~]$
```

3.2.3 启动Spark集群和Spark SQL CLI

通过如下命令启动Spark集群和Spark SQL CLI：

```
$cd /app/hadoop/spark-1.1.0
$sbin/start-all.sh
$bin/spark-sql --master spark://hadoop1:7077 --executor-memory 1g
```

在集群监控页面可以看到启动了SparkSQL应用程序：

这时就可以使用HQL语句对Hive数据进行查询，另外可以使用COMMAND，如使用set进行设置参数：默认情况下，SparkSQL Shuffle的时候是200个partition，可以使用如下命令修改该参数：

```
SET spark.sql.shuffle.partitions=20;
```

运行同一个查询语句，参数改变后，Task（partition）的数量就由200变成了20。

18	mapPartitions at Exchange.scala:48	+details	2014/09/14 22:13:53	0.7 s	20/20		1330.7 KB	139.5 KB
17	mapPartitions at Exchange.scala:48	+details	2014/09/14 22:13:52	0.7 s	20/20		1392.6 KB	1952. KB
16	mapPartitions at Exchange.scala:48	+details	2014/09/14 22:13:50	2 s	2/2	11.7 MB		1675. KB
15	mapPartitions at Exchange.scala:48	+details	2014/09/14 22:13:50	1 s	2/2	608.7 KB		317.6 KB
14	mapPartitions at Exchange.scala:48	+details	2014/09/14 22:13:50	0.9 s	2/2	171.8 KB		51.3 KB
6	collect at HiveContext.scala:415	+details	2014/09/14 22:12:52	0.1 s	8/8		299.0 B	
12	mapPartitions at Exchange.scala:71	+details	2014/09/14 22:12:50	2 s	200/200		144.3 KB	419.0 B
0	RangePartitioner at Exchange.scala:79	+details	2014/09/14 22:12:48	2 s	200/200		145.5 KB	

3.2.4 命令参数

通过bin/spark-sql --help可以查看CLI命令参数：

```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 spark-1.1.0]$ bin/spark-sql --help
Usage: ./bin/spark-sql [options] [cli option]
Spark assembly has been built with Hive, including Datanucleus jars on classpath
Options:
  --master MASTER_URL         spark://host:port, mesos://host:port, yarn, or local.
  --deploy-mode DEPLOY_MODE   Whether to launch the driver program locally ("client") or
                              on one of the worker machines inside the cluster ("cluster")
                              (Default: client).
  --class CLASS_NAME          Your application's main class (for Java / Scala apps).
  --name NAME                 A name of your application.
  --jars JARS                 Comma-separated list of local jars to include on the driver
                              and executor classpaths.
  --py-files PY_FILES         Comma-separated list of .zip, .egg, or .py files to place
                              on the PYTHONPATH for Python apps.
  --files FILES               Comma-separated list of files to be placed in the working
                              directory of each executor.

  --conf PROP=VALUE           Arbitrary Spark configuration property.
  --properties-file FILE      Path to a file from which to load extra properties. If not
                              specified, this will look for conf/spark-defaults.conf.

  --driver-memory MEM         Memory for driver (e.g. 1000M, 2G) (Default: 512M).
  --driver-java-options       Extra Java options to pass to the driver.
  --driver-library-path       Extra library path entries to pass to the driver.
  --driver-class-path         Extra class path entries to pass to the driver. Note that
                              jars added with --jars are automatically included in the
                              classpath.

  --executor-memory MEM       Memory per executor (e.g. 1000M, 2G) (Default: 1G).

  --help, -h                  Show this help message and exit
  --verbose, -v               Print additional debug output

 Spark standalone with cluster deploy mode only:
  --driver-cores NUM          Cores for driver (Default: 1).
  --supervise                 If given, restarts the driver on failure.
```

```
CLI options:
15/07/30 22:11:29 INFO Configuration.deprecation: mapred.input.dir.recursive is deprecated. Instead, use mapreduc
15/07/30 22:11:29 INFO Configuration.deprecation: mapred.max.split.size is deprecated. Instead, use mapreduce.inp
15/07/30 22:11:29 INFO Configuration.deprecation: mapred.min.split.size is deprecated. Instead, use mapreduce.inp
15/07/30 22:11:29 INFO Configuration.deprecation: mapred.min.split.size.per.rack is deprecated. Instead, use mapr
15/07/30 22:11:29 INFO Configuration.deprecation: mapred.min.split.size.per.node is deprecated. Instead, use mapr
15/07/30 22:11:29 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.r
15/07/30 22:11:29 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instea
 -d,--define <key=value>          Variable subsitution to apply to hive
                                  commands. e.g. -d A=B or --define A=B
    --database <databasename>     Specify the database to use
 -e <quoted-query-string>         SQL from command line
 -f <filename>                    SQL from files
 -h <hostname>                    connecting to Hive Server on remote host
    --hiveconf <property=value>   Use value for given property
    --hivevar <key=value>         Variable subsitution to apply to hive
                                  commands. e.g. --hivevar A=B
 -i <filename>                    Initialization SQL file
 -p <port>                        connecting to Hive Server on port number
 -S,--silent                      Silent mode in interactive shell
 -v,--verbose                     Verbose mode (echo executed SQL to the
                                  console)
```

其中[options] 是CLI启动一个SparkSQL应用程序的参数，如果不设置--master的话，将在启动spark-sql的机器以local方式运行，只能通过http://机器名:4040进行监控；这部分参数，可以参照Spark1.0.0 应用程序部署工具spark-submit 的参数。

[cli option]是CLI的参数，通过这些参数CLI可以直接运行SQL文件、进入命令行运行SQL命令等等，类似以前的Shark的用法。需要注意的是CLI不是使用JDBC连接，所以不能连接到ThriftServer；但可以配置conf/hive-site.xml连接到Hive的Metastore，然后对Hive数据进行查询。

3.3 实战Spark SQL CLI

3.3.1 获取订单每年的销售单数、销售总额

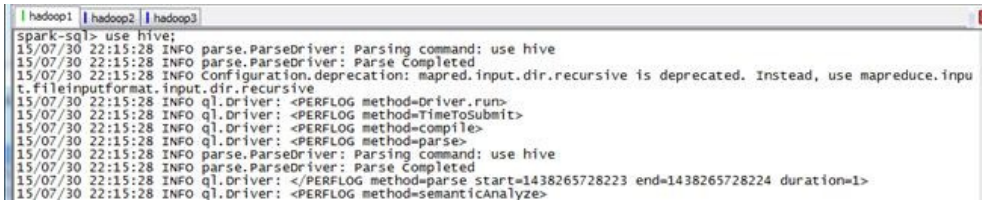
第一步 设置任务个数，在这里修改为20个

spark-sql>SET spark.sql.shuffle.partitions=20;

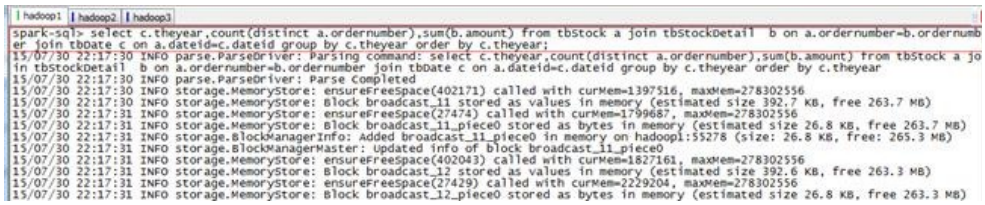
```
hadoop1 | hadoop2 | hadoop3
spark-sql> SET spark.sql.shuffle.partitions=20;
SET spark.sql.shuffle.partitions=20
spark.sql.shuffle.partitions=20
Time taken: 0.089 seconds
15/07/30 22:14:35 INFO CliDriver: Time taken: 0.089 seconds
spark-sql>
```

第二步 运行SQL语句

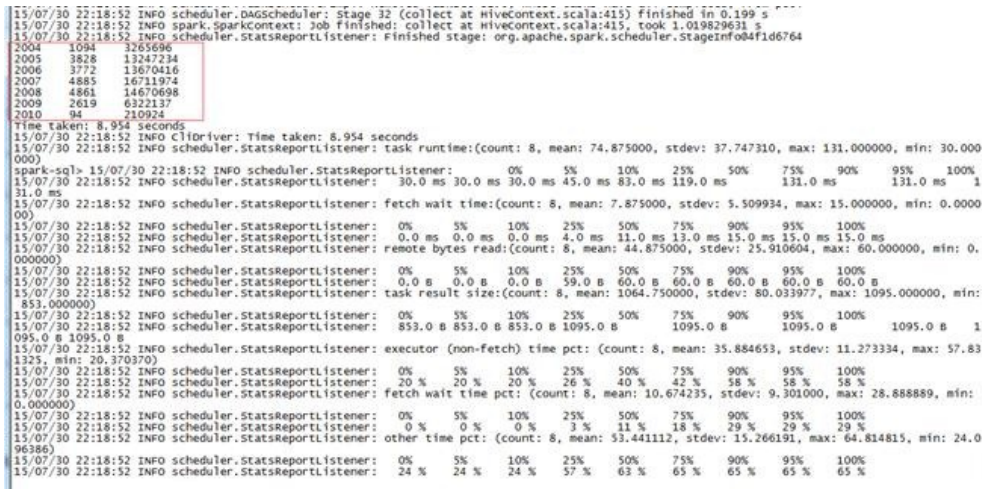
spark-sql>use hive;

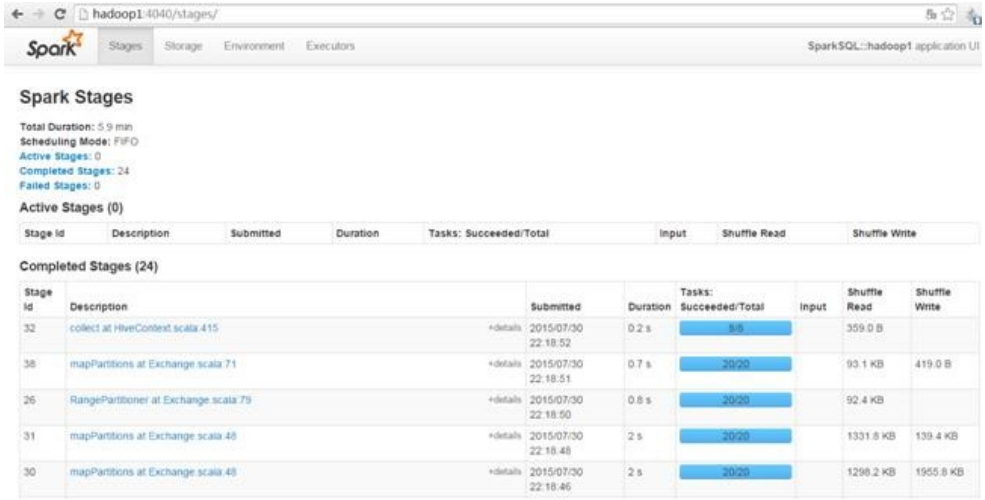


spark-sql>select c.theyear,count(distinct a.ordernumber),sum(b.amount) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear;



第三步 查看运行结果

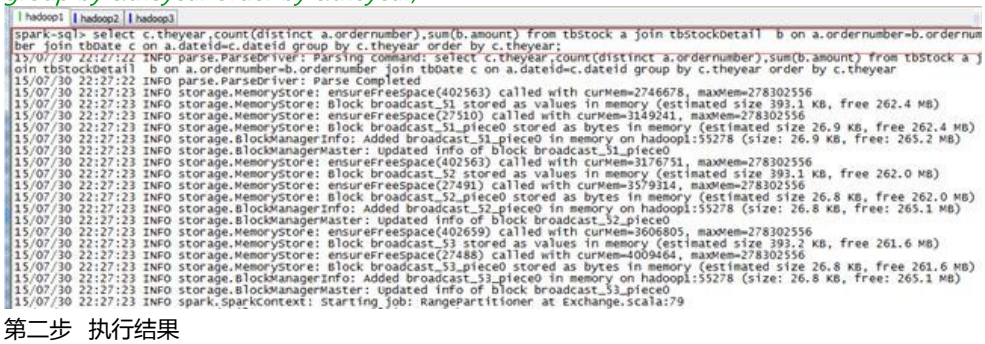




3.3.2 计算所有订单每年的总金额

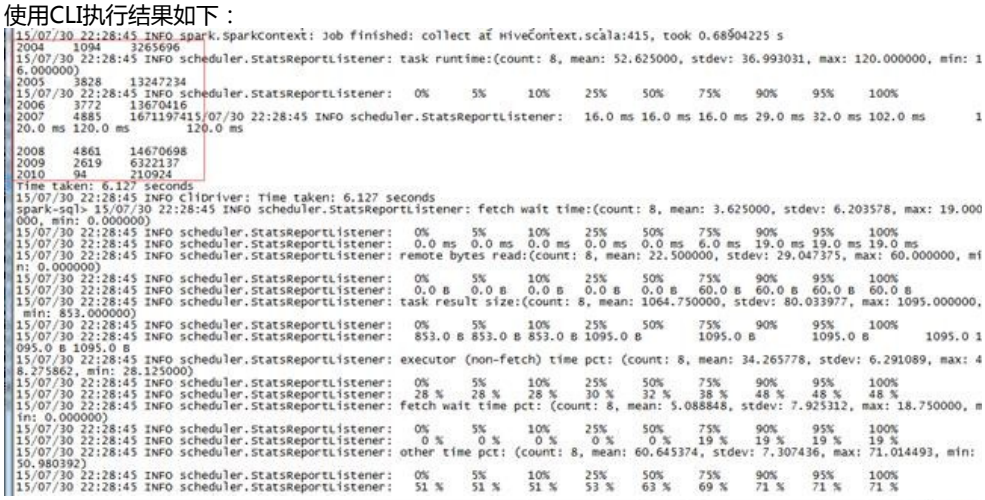
第一步 执行SQL语句

spark-sql>select c.theyear,count(distinct a.ordernumber),sum(b.amount) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear;



第二步 执行结果

使用CLI执行结果如下：

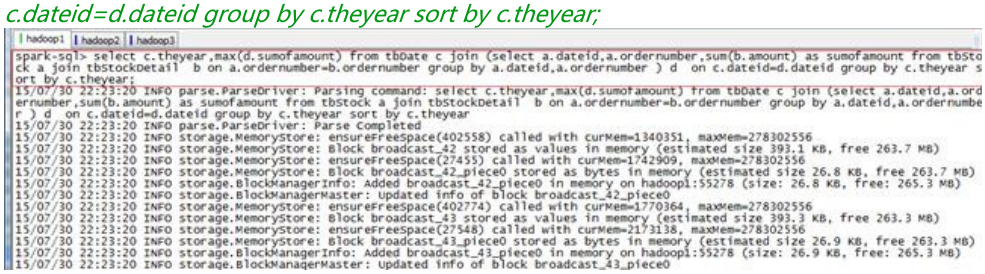


Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
57	collect at HiveContext.scala:415	+details	2015/07/30 22:27:30	0.2 s	8/8		179.0 B	
63	mapPartitions at Exchange.scala:71	+details	2015/07/30 22:27:30	0.7 s	20/20		93.3 KB	419.0 B
51	RangePartitioner at Exchange.scala:79	+details	2015/07/30 22:27:28	0.8 s	20/20		93.8 KB	
56	mapPartitions at Exchange.scala:48	+details	2015/07/30 22:27:26	2 s	20/20		1299.2 KB	139.4 KB
55	mapPartitions at Exchange.scala:48	+details	2015/07/30 22:27:25	1 s	20/20		1359.3 KB	1895.9 K
54	mapPartitions at Exchange.scala:48	+details	2015/07/30 22:27:23	2 s	2/2	11.4 MB		1675.0 K
53	mapPartitions at Exchange.scala:48	+details	2015/07/30 22:27:23	0.9 s	2/2	588.0 KB		333.7 KB
52	mapPartitions at Exchange.scala:48	+details	2015/07/30 22:27:23	0.8 s	2/2	167.5 KB		51.3 KB

3.3.3 计算所有订单每年最大金额订单的销售额

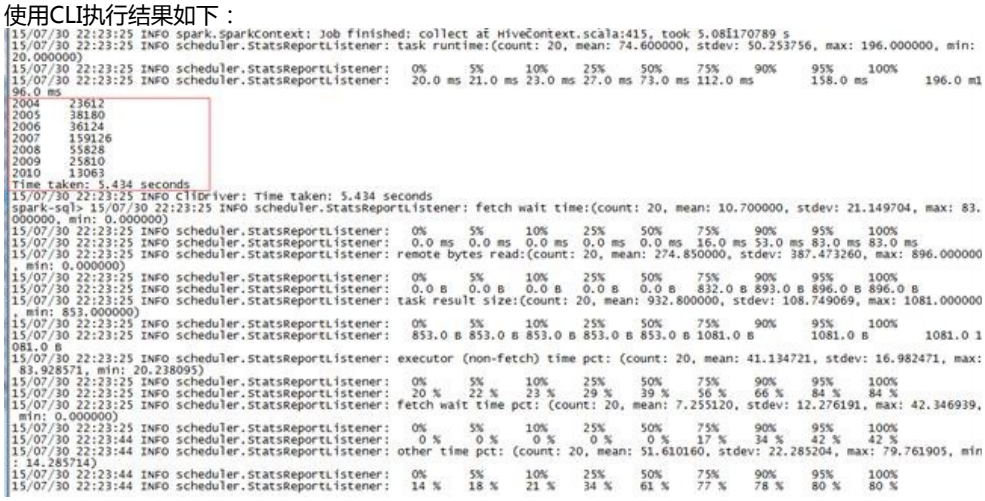
第一步 执行SQL语句

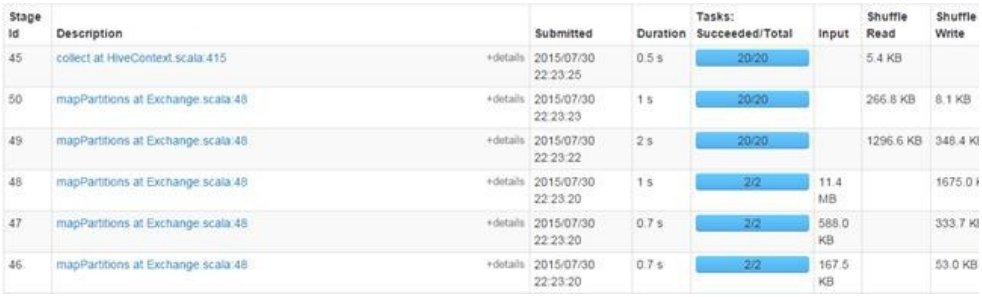
spark-sql>select c.theyear,max(d.sumofamount) from tbDate c join (select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber) d on c.dateid=d.dateid group by c.theyear sort by c.theyear;



第二步 执行结果

使用CLI执行结果如下：





Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
45	collect at HiveContext.scala:415 +details	2015/07/30 22:23:25	0.5 s	20/20		5.4 KB	
50	mapPartitions at Exchange.scala:48 +details	2015/07/30 22:23:23	1 s	20/20		266.8 KB	8.1 KB
49	mapPartitions at Exchange.scala:48 +details	2015/07/30 22:23:22	2 s	20/20		1296.6 KB	348.4 KB
48	mapPartitions at Exchange.scala:48 +details	2015/07/30 22:23:20	1 s	2/2	11.4 MB		1675.0 KB
47	mapPartitions at Exchange.scala:48 +details	2015/07/30 22:23:20	0.7 s	2/2	588.0 KB		333.7 KB
46	mapPartitions at Exchange.scala:48 +details	2015/07/30 22:23:20	0.7 s	2/2	167.5 KB		53.0 KB

4、Spark Thrift Server

ThriftServer是一个JDBC/ODBC接口，用户可以通过JDBC/ODBC连接ThriftServer来访问SparkSQL的数据。ThriftServer在启动的时候，会启动了一个SparkSQL的应用程序，而通过JDBC/ODBC连接进来的客户端共同分享这个SparkSQL应用程序的资源，也就是说不同的用户之间可以共享数据；ThriftServer启动时还开启一个侦听器，等待JDBC客户端的连接和提交查询。所以，在配置ThriftServer的时候，至少要配置ThriftServer的主机名和端口，如果要使用Hive数据的话，还要提供Hive Metastore的uris。

【注】Spark CLI和Spark Thrift Server实验环境为第二课《Spark编译与部署（下）--Spark编译安装》所搭建

4.1 配置并启动

4.1.1 创建并配置hive-site.xml

第一步 创建hive-site.xml配置文件

在$SPARK_HOME/conf目录下修改hive-site.xml配置文件（如果在Spark SQL CLI中已经添加，可以省略）：

```
$cd /app/hadoop/spark-1.1.0/conf
$sudo vi hive-site.xml
```

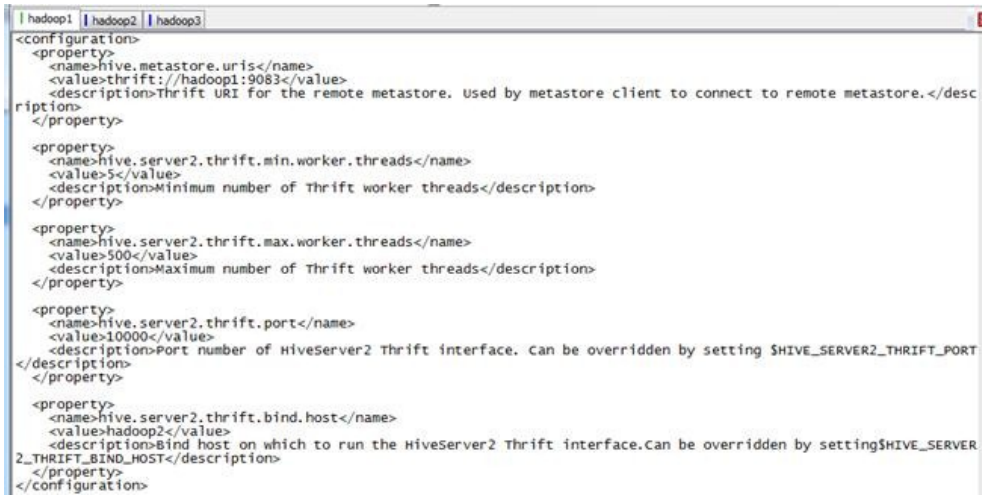
```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 ~]$ cd /app/hadoop/spark-1.1.0/conf
[hadoop@hadoop1 conf]$ ls
fairscheduler.xml.template  log4j.properties.template    spark-defaults.conf.template
hive-site-bak.xml           metrics.properties.template  spark-env.sh
hive-site.xml               slaves                       spark-env.sh.template
[hadoop@hadoop1 conf]$ sudo vi hive-site.xml
```

第二步 修改配置文件

设置hadoop1为Metastore服务器，hadoop2为Thrift Server服务器，配置内容如下：

```
<configuration>
  <property>
    <name>hive.metastore.uris</name>
    <value>thrift://hadoop1:9083</value>
    <description>Thrift URI for the remote metastore. Used by metastore client to connect
      to remote metastore.</description>
  </property>

  <property>
    <name>hive.server2.thrift.min.worker.threads</name>
    <value>5</value>
    <description>Minimum number of Thrift worker threads</description>
  </property>

  <property>
    <name>hive.server2.thrift.max.worker.threads</name>
    <value>500</value>
    <description>Maximum number of Thrift worker threads</description>
  </property>

  <property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
    <description>Port number of HiveServer2 Thrift interface. Can be overridden by setting
      $HIVE_SERVER2_THRIFT_PORT</description>
  </property>

  <property>
```

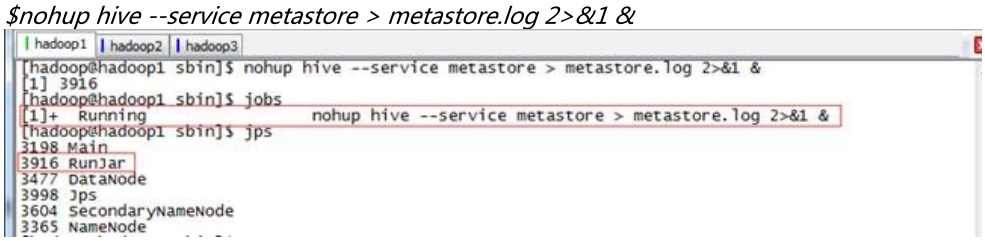
```
    <name>hive.server2.thrift.bind.host</name>
    <value>hadoop2</value>
    <description>Bind host on which to run the HiveServer2 Thrift interface.Can be
      overridden by setting$HIVE_SERVER2_THRIFT_BIND_HOST</description>
  </property>
</configuration>
```



4.1.2 启动Hive

在hadoop1节点中，在后台启动Hive Metastore（如果数据存放在HDFS文件系统，还需要启动Hadoop的HDFS）：

```
$nohup hive --service metastore > metastore.log 2>&1 &
```



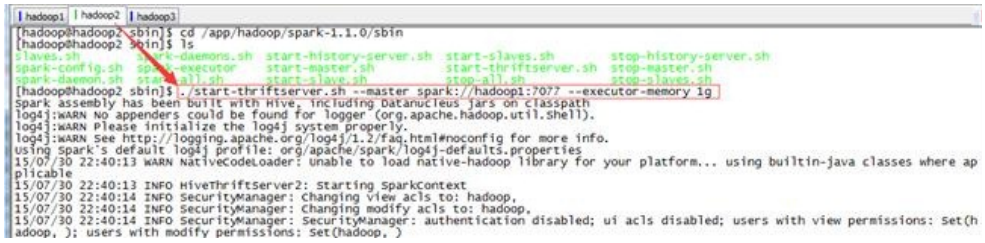
4.1.3 启动Spark集群和Thrift Server

在hadoop1节点启动Spark集群

```
$cd /app/hadoop/spark-1.1.0/sbin
$./start-all.sh
```

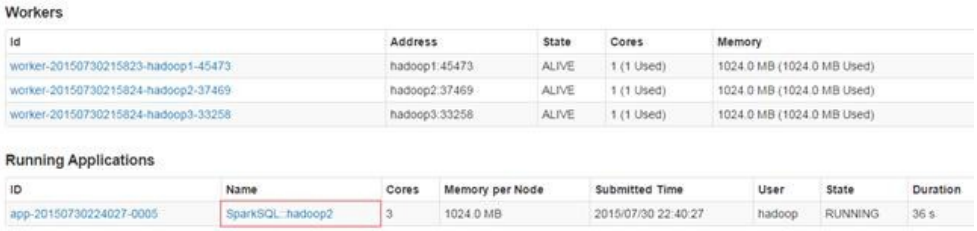
在hadoop2节点上进入SPARK_HOME/sbin目录，使用如下命令启动Thrift Server

```
$cd /app/hadoop/spark-1.1.0/sbin
$./start-thriftserver.sh --master spark://hadoop1:7077 --executor-memory 1g
```



注意：Thrift Server需要按照配置在hadoop2启动！

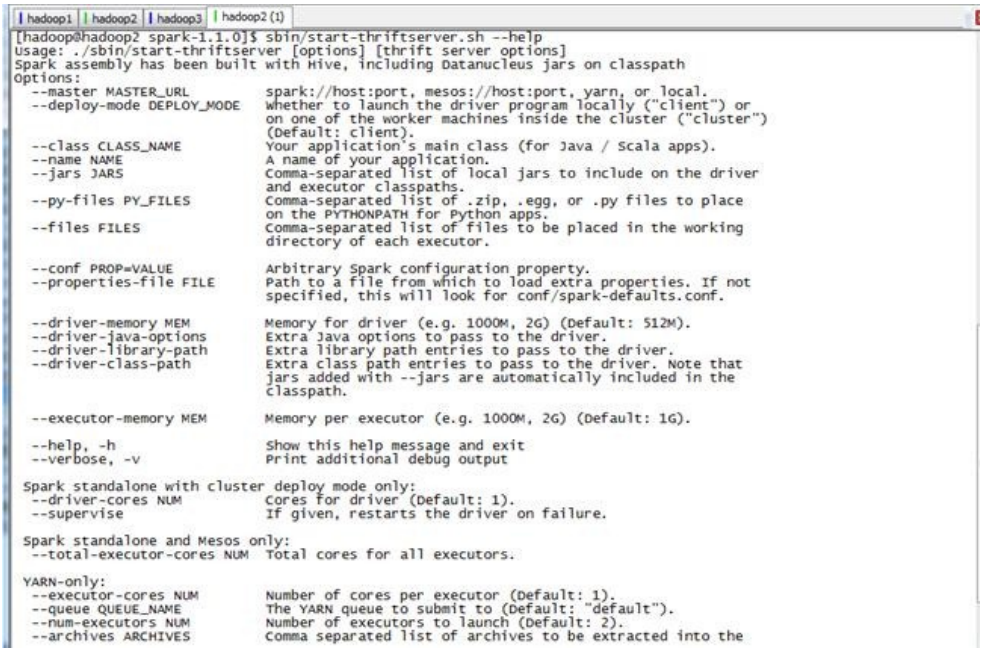
在集群监控页面可以看到启动了SparkSQL应用程序：



4.1.4 命令参数

使用sbin/start-thriftserver.sh --help可以查看ThriftServer的命令参数：

$sbin/start-thriftserver.sh --help Usage: ./sbin/start-thriftserver [options] [thrift server options]
Thrift server options: Use value for given property



其中[options] 是Thrift Server启动一个SparkSQL应用程序的参数，如果不设置--master的话，将在启动Thrift Server的机器以local方式运行，只能通过http://机器名:4040进行监控；这部分参数，可以参照Spark1.0.0 应用程序部署工具spark-submit 的参数。在集群中提供Thrift Server的话，一定要配置master、executor-memory等参数。

[thrift server options]是Thrift Server的参数，可以使用-dproperty=value的格式来定义；在实际应用上，因为参数比较多，通常使用conf/hive-site.xml配置。

4.2 实战Thrift Server

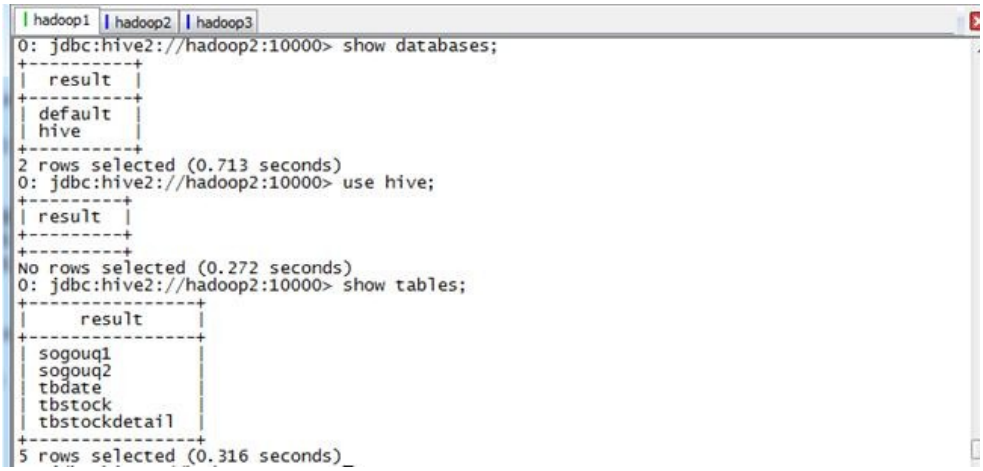
4.2.1 远程客户端连接

可以在任意节点启动bin/beeline，用!connect jdbc:hive2://hadoop2:10000连接ThriftServer，因为没有采用权限管理，所以用户名用运行bin/beeline的用户hadoop，密码为空：

$cd /app/hadoop/spark-1.1.0/bin
$./beeline
beeline>!connect jdbc:hive2://hadoop2:10000

4.2.2 基本操作

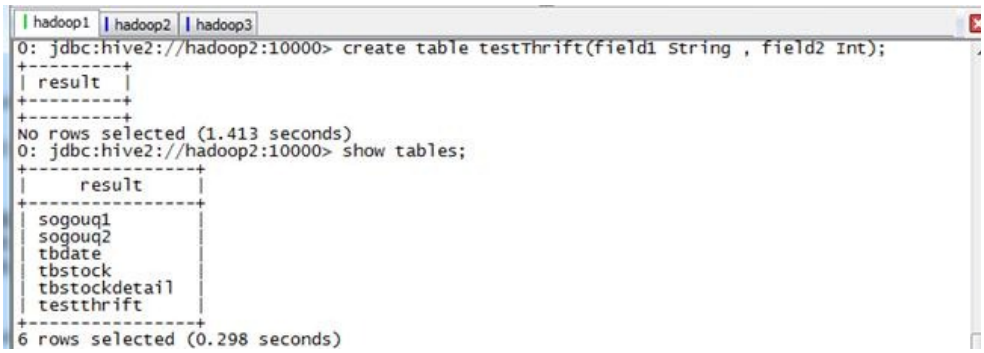
第一步 显示hive数据库所有表
beeline>show database;
beeline>use hive;
beeline>show tables;



第二步 创建表testThrift

beeline>create table testThrift(field1 String , field2 Int);

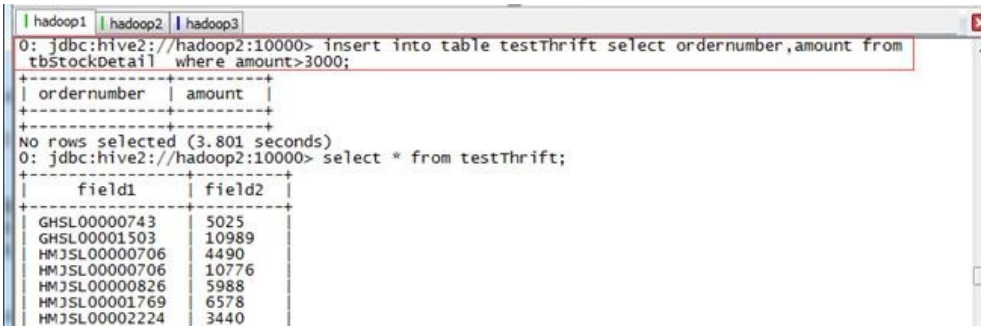
beeline>show tables;



第三步 把tbStockDetail表中金额大于3000插入到testThrift表中

beeline>insert into table testThrift select ordernumber,amount from tbStockDetail where amount>3000;

*beeline>select * from testThrift;*



第四步 重新创建testThrift表中，把年度最大订单插入该表中

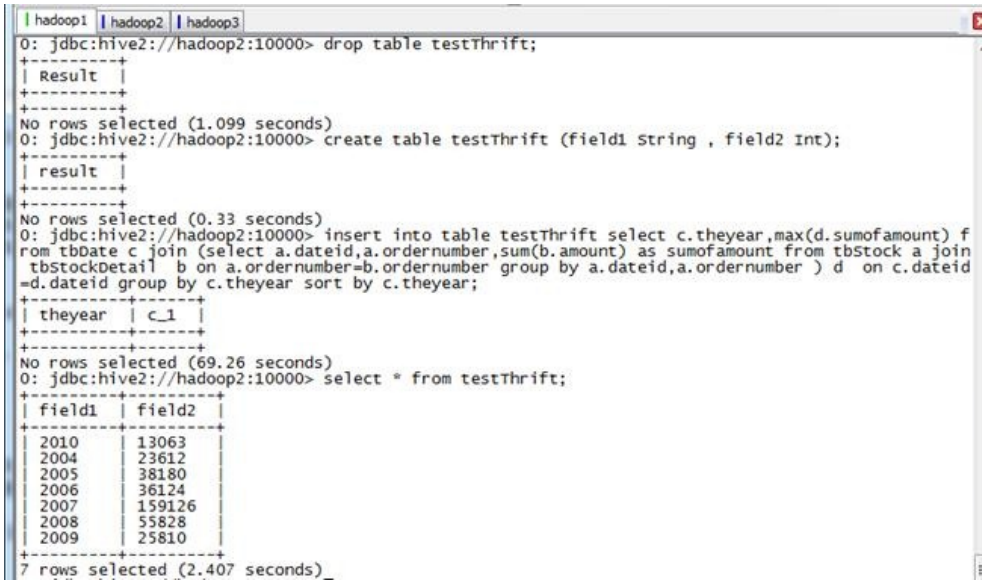
beeline>drop table testThrift;

beeline>create table testThrift (field1 String , field2 Int);

beeline>insert into table testThrift select c.theyear,max(d.sumofamount) from tbDate c joi

(select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber) d on c.dateid=d.dateid group by c.theyear sort by c.theyear;

*beeline>select * from testThrift;*

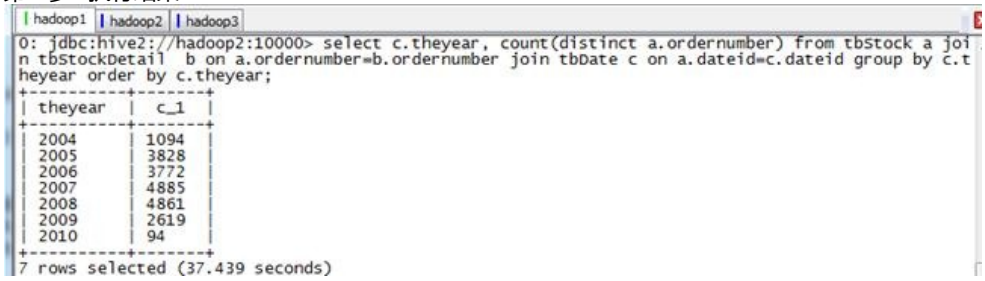


4.2.3 计算所有订单每年的订单数

第一步 执行SQL语句

spark-sql>select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDeta
b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by
c.theyear order by c.theyear;

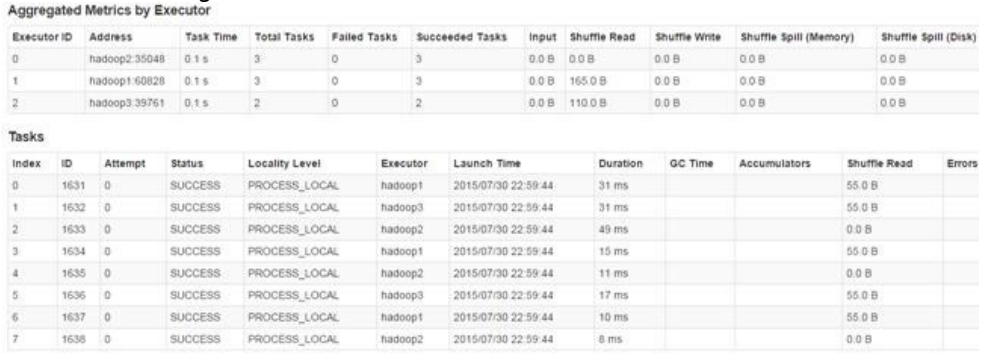
第二步 执行结果



Stage监控页面：

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuff Write
28	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear collect at SparkSQLOperationManager.scala:191 +details	2015/07/30 22:59:44	0.1 s	8/8		275.0 B	
34	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear mapPartitions at Exchange.scala:71 +details	2015/07/30 22:59:41	3 s	200/200		143.5 KB	365.0
22	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear RangePartitioner at Exchange.scala:79 +details	2015/07/30 22:59:38	3 s	200/200		143.0 KB	
27	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear mapPartitions at Exchange.scala:48 +details	2015/07/30 22:59:23	15 s	200/200		1470.0 KB	214.7 KB
26	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear mapPartitions at Exchange.scala:48 +details	2015/07/30 22:59:09	13 s	200/200		988.6 KB	2.1 M
25	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear mapPartitions at Exchange.scala:48 +details	2015/07/30 22:59:07	2 s	2/2	11.4 MB		1058. KB
24	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear mapPartitions at Exchange.scala:48 +details	2015/07/30 22:59:07	1 s	2/2	588.0 KB		413.0 KB
23	select c.theyear, count(distinct a.ordernumber) from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear order by c.theyear mapPartitions at Exchange.scala:48 +details	2015/07/30 22:59:07	0.7 s	2/2	167.5 KB		87.3 KB

查看Details for Stage 28



Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
0	hadoop2:35048	0.1 s	3	0	3	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B
1	hadoop1:60828	0.1 s	3	0	3	0.0 B	165.0 B	0.0 B	0.0 B	0.0 B
2	hadoop3:39761	0.1 s	2	0	2	0.0 B	110.0 B	0.0 B	0.0 B	0.0 B

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Shuffle Read	Errors
0	1631	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/30 22:59:44	31 ms			55.0 B	
1	1632	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/30 22:59:44	31 ms			55.0 B	
2	1633	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/30 22:59:44	49 ms			0.0 B	
3	1634	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/30 22:59:44	15 ms			55.0 B	
4	1635	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/30 22:59:44	11 ms			0.0 B	
5	1636	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/30 22:59:44	17 ms			55.0 B	
6	1637	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/30 22:59:44	10 ms			55.0 B	
7	1638	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/30 22:59:44	8 ms			0.0 B	

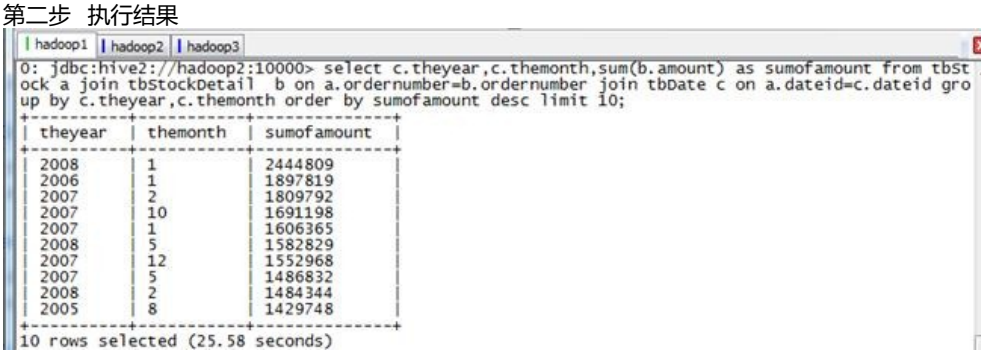
4.2.4 计算所有订单月销售额前十名

第一步 执行SQL语句

spark-sql>select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a joi
tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid

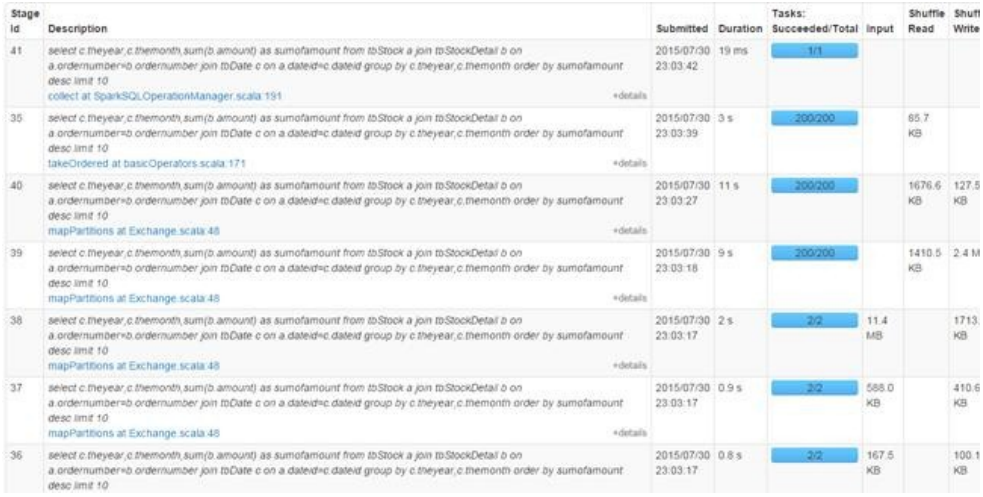
group by c.theyear,c.themonth order by sumofamount desc limit 10;

第二步 执行结果



```
hadoop1 | hadoop2 | hadoop3
0: jdbc:hive2://hadoop2:10000> select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbSt
ock a join tbStockDetail  b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid gro
up by c.theyear,c.themonth order by sumofamount desc limit 10;
+-----------+------------+---------------+
| theyear   | themonth   | sumofamount   |
+-----------+------------+---------------+
| 2008      | 1          | 2444809       |
| 2006      | 1          | 1897819       |
| 2007      | 2          | 1809792       |
| 2007      | 10         | 1691198       |
| 2007      | 1          | 1606365       |
| 2008      | 5          | 1582829       |
| 2007      | 12         | 1552968       |
| 2007      | 5          | 1486832       |
| 2008      | 2          | 1484344       |
| 2005      | 8          | 1429748       |
+-----------+------------+---------------+
10 rows selected (25.58 seconds)
```

Stage监控页面：



Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuff Write
41	select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,c.themonth order by sumofamount desc limit 10 collect at SparkSQLOperationManager.scala:191 +details	2015/07/30 23:03:42	19 ms	1/1			
35	select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,c.themonth order by sumofamount desc limit 10 takeOrdered at basicOperators.scala:171 +details	2015/07/30 23:03:39	3 s	200/200		65.7 KB	
40	select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,c.themonth order by sumofamount desc limit 10 mapPartitions at Exchange.scala:48 +details	2015/07/30 23:03:27	11 s	200/200		1676.6 KB	127.5 KB
39	select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,c.themonth order by sumofamount desc limit 10 mapPartitions at Exchange.scala:48 +details	2015/07/30 23:03:18	9 s	200/200		1410.5 KB	2.4 M
38	select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,c.themonth order by sumofamount desc limit 10 mapPartitions at Exchange.scala:48 +details	2015/07/30 23:03:17	2 s	2/2	11.4 MB		1713. KB
37	select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,c.themonth order by sumofamount desc limit 10 mapPartitions at Exchange.scala:48 +details	2015/07/30 23:03:17	0.9 s	2/2	588.0 KB		410.6 KB
36	select c.theyear,c.themonth,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,c.themonth order by sumofamount desc limit 10	2015/07/30 23:03:17	0.8 s	2/2	167.5 KB		100.1 KB

在其第一个Task中，从本地读入数据

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Dis
1	hadoop1:60828	0.6 s	1	0	1	83.7 KB	0.0 B	49.6 KB	0.0 B	0.0 B
2	hadoop3:39761	0.8 s	1	0	1	83.7 KB	0.0 B	50.6 KB	0.0 B	0.0 B

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Input	Write Time	Shuffle Write	Erro
0	1639	0	SUCCESS	NODE_LOCAL	hadoop1	2015/07/30 23:03:17	0.5 s			83.7 KB (hadoop)	6 ms	49.6 KB	
1	1640	0	SUCCESS	NODE_LOCAL	hadoop3	2015/07/30 23:03:17	0.7 s	30 ms		83.7 KB (hadoop)	6 ms	50.6 KB	

在后面的Task是从内存中获取数据

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
0	hadoop2:35048	3 s	60	0	60	0.0 B	28.8 KB	0.0 B	0.0 B	0.0 B
1	hadoop1:60828	3 s	70	0	70	0.0 B	28.2 KB	0.0 B	0.0 B	0.0 B
2	hadoop3:39761	3 s	70	0	70	0.0 B	28.7 KB	0.0 B	0.0 B	0.0 B

Tasks

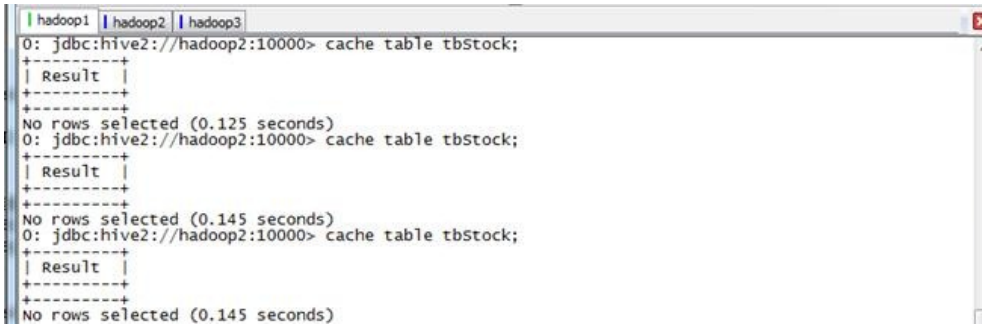
Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Shuffle Read	Errors
0	2045	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/30 23:03:39	0.2 s			0.0 B	
2	2047	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/30 23:03:39	0.2 s			0.0 B	
1	2046	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/30 23:03:39	0.2 s			0.0 B	
3	2048	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/30 23:03:39	3 ms			0.0 B	
4	2049	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/30 23:03:39	6 ms			0.0 B	
6	2051	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/30 23:03:39	75 ms			1330.0 B	
5	2050	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/30 23:03:39	73 ms			1400.0 B	
7	2052	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/30 23:03:39	67 ms			1470.0 B	
8	2053	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/30 23:03:39	61 ms			1469.0 B	

4.2.5 缓存表数据

第一步 缓存数据

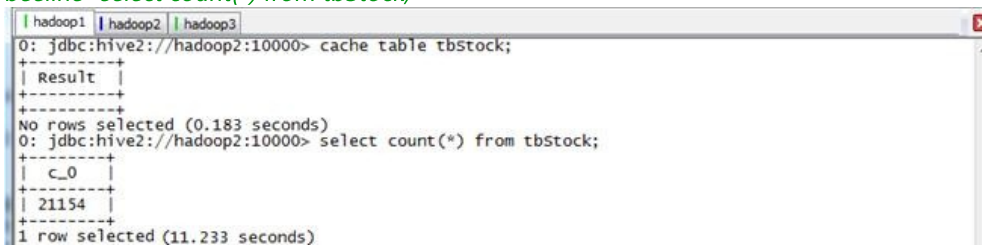
beeline>cache table tbStock;

beeline>select count(*) from tbStock;



第二步　运行4.2.4中的“计算所有订单月销售额前十名”

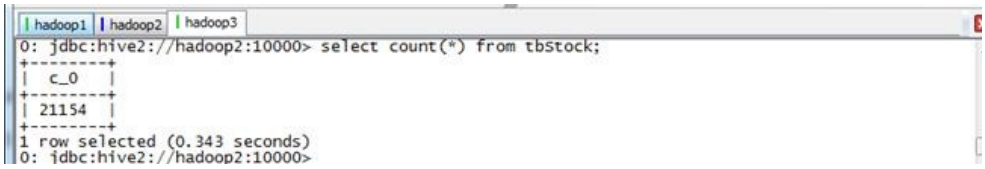
```
beeline>select count(*) from tbStock;
```



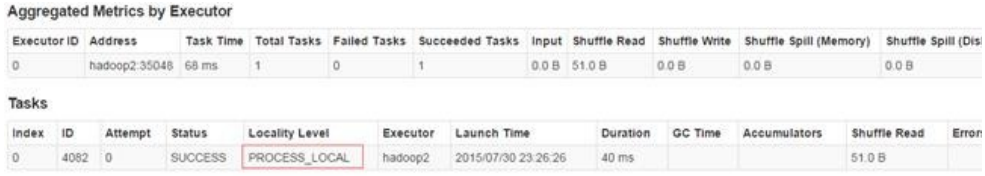
本次计算划给11.233秒，查看webUI，数据已经缓存，缓存率为100%：

第三步　在另外节点再次运行

在hadoop3节点启动bin/beeline，用!connect jdbc:hive2://hadoop2:10000连接ThriftServer，然后直接运行对tbStock计数（注意没有进行数据库的切换）：



用时0.343秒，再查看webUI中的stage：



Locality Level是PROCESS，显然是使用了缓存表。

从上可以看出，ThriftServer可以连接多个JDBC/ODBC客户端，并相互之间可以共享数据。顺便提一句，ThriftServer启动后处于监听状态，用户可以使用ctrl+c退出ThriftServer；而beeline的退出使用!q命令。

4.2.6 在IDEA中JDBC访问

有了ThriftServer，开发人员可以非常方便的使用JDBC/ODBC来访问SparkSQL。下面是一个scala代码，查询表tbStockDetail，返回amount>3000的单据号和交易金额：

第一步　在IDEA创建class6包和类JDBCofSparkSQL

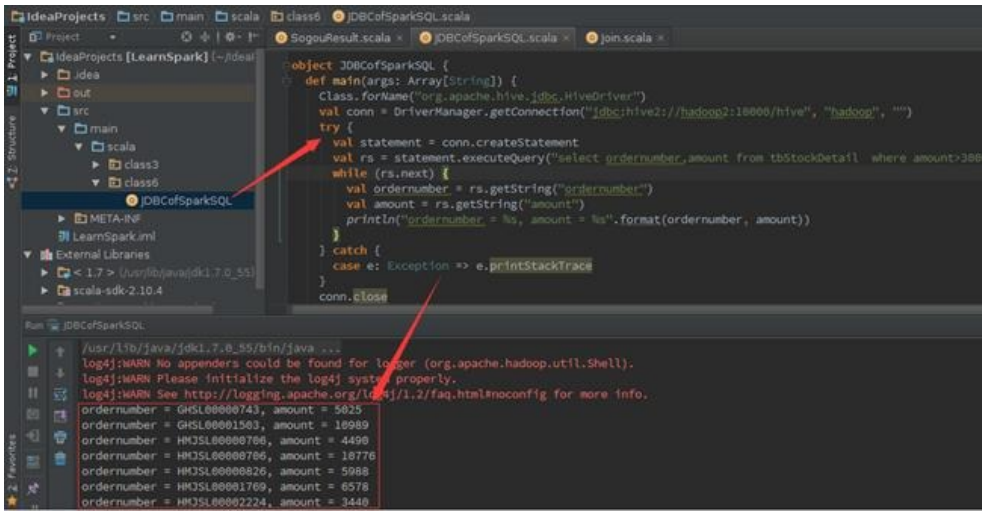
参见《Spark编程模型（下）--IDEA搭建及实战》在IDEA中创建class6包并新建类JDBCofSparkSQL。该类中查询tbStockDetail金额大于3000的订单：

```
package class6
import java.sql.DriverManager

object JDBCofSparkSQL {
  def main(args: Array[String]) {
    Class.forName("org.apache.hive.jdbc.HiveDriver")
    val conn = DriverManager.getConnection("jdbc:hive2://hadoop2:10000/hive", "hadoop"
"")
```

```
    try {
      val statement = conn.createStatement
          val rs = statement.executeQuery("select ordernumber,amount from tbStockDetail
           where amount>3000")
      while (rs.next) {
        val ordernumber = rs.getString("ordernumber")
        val amount = rs.getString("amount")
        println("ordernumber = %s, amount = %s".format(ordernumber, amount))
      }
    } catch {
      case e: Exception => e.printStackTrace
    }
    conn.close
  }
}
```

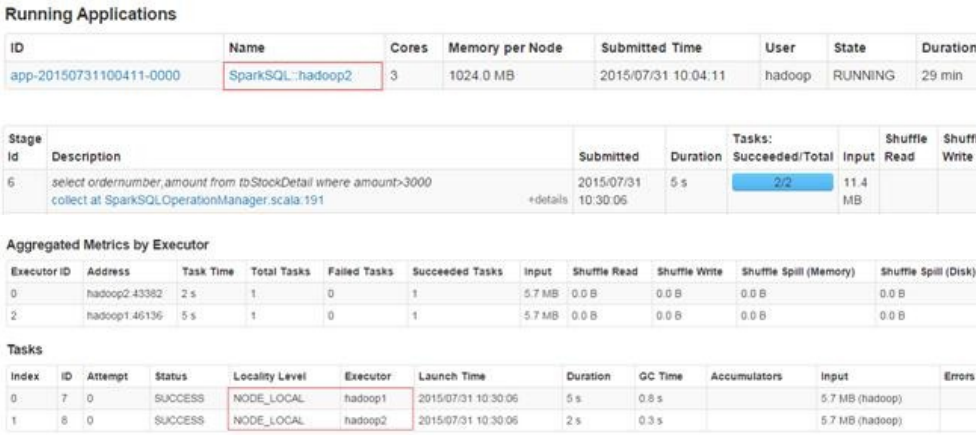
第二步　查看运行结果

在IDEA中可以观察到，在运行日志窗口中没有运行过程的日志，只显示查询结果



第三步　查看监控结果

从Spark监控界面中观察到，该Job有一个编号为6的Stage，该Stage有2个Task，分别运行在hadoop1和hadoop2节点，获取数据为NODE_LOCAL方式。



Running Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20150731100411-0000	SparkSQL::hadoop2	3	1024.0 MB	2015/07/31 10:04:11	hadoop	RUNNING	29 min

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuff Write
6	select ordernumber,amount from tbStockDetail where amount>3000 collect at SparkSQLOperationManager.scala:191 +details	2015/07/31 10:30:06	5 s	2/2	11.4 MB		

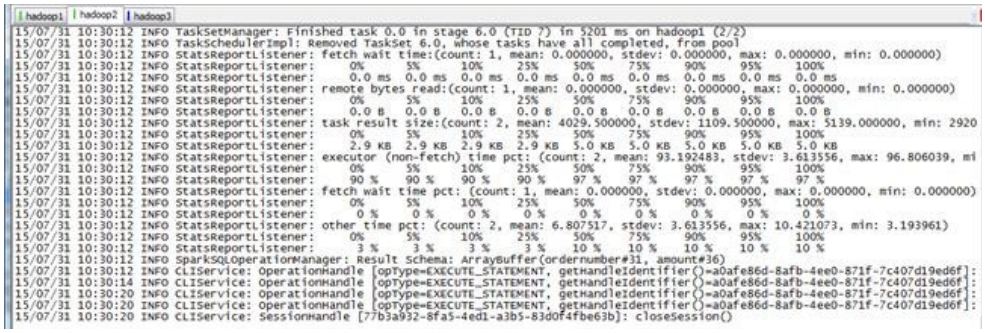
Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
0	hadoop2:43382	2 s	1	0	1	5.7 MB	0.0 B	0.0 B	0.0 B	0.0 B
2	hadoop1:46136	5 s	1	0	1	5.7 MB	0.0 B	0.0 B	0.0 B	0.0 B

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Input	Errors
0	7	0	SUCCESS	NODE_LOCAL	hadoop1	2015/07/31 10:30:06	5 s	0.8 s		5.7 MB (hadoop)	
1	8	0	SUCCESS	NODE_LOCAL	hadoop2	2015/07/31 10:30:06	2 s	0.3 s		5.7 MB (hadoop)	

在hadoop2中观察Thrift Server运行日志如下：



分类: 20.Spark入门实战系列

shishanyuan
关注 - 16
粉丝 - 659
13 0
荣誉：推荐博客
+加关注

« 上一篇：Spark入门实战系列--5.Hive（下）--Hive实战
» 下一篇：Spark入门实战系列--6.SparkSQL（中）--深入了解SparkSQL运行计划及调优

posted @ 2015-08-26 09:03 shishanyuan 阅读(97999) 评论(28) 编辑 收藏

发表评论

#1楼 2015-10-27 11:39 | tzou

请问一下，schemardd和dataframe有什么区别？
1.3.0之后的版本是不是没有schemardd的概念了？
谢谢！

支持(0) 反对(

#2楼[楼主] 2015-10-29 09:36 | shishanyuan

@ tzou
在Spark 1.3.0以Spark SQL原有的SchemaRDD为蓝本，引入了Spark DataFrame API，具体他们之间的区别可参见Databricks公司连城写的文章
http://code.csdn.net/news/2824958

支持(0) 反对(

#3楼 2015-11-02 22:16 | 张平a

hive2是啥东东？

支持(0) 反对(

#4楼[楼主] 2015-11-05 10:28 | shishanyuan

@ 张平a
hiveserver2是hiveserver的改进版本，相比而言，hiveserver2更加稳定，支持的功能更多，hiveserver 和 hiverver2 在使用JDBC连接方面有两个不同的地方：
1.驱动类：org.apache.hadoop.hive.jdbc.HiveDriver –> org.apache.hive.jdbc.HiveDriver
2.URL：jdbc:hive://localhost:10000/default –> jdbc:hive2://localhost:10000/default

支持(0) 反对(

#5楼 2015-11-06 08:52 | 张平a

@ shishanyuan
老师，您好：
按照您的思路一路做下来都很正常，现在唯独卡有一个问题上研究一天多也没解决掉：jdbc连接ThriftServer。我的境跟您略有不同，1、我是单台服务器；2、我用的是cdh530版本（spark重新编译，hadoop和hive直接下得gz包）；
前提：1、spark/conf/hive-site.xml已经设置；2、metastore已经启动；3、通过日志查看thriftserver启动一切正常；问题是卡在beeline>!connect jdbc:hive2://hostname:10000时一直在报一个错误：Error: Could not establish connection to jdbc:hive2://hostname:10000: Internal error processing OpenSession (state=08S01,code=0)上；以为是mysql 的驱动包没有引入，引入 到spark/lib下问题依旧，而且我发现在此之前就有了异常执行./beeline后找不到hive版本号（一直是三个问号）：Beeline version ??? by Apache Hive；我现在怀疑是不版本问题，但是我都是下得cdh同一版本，按理说不应该是版本问题；

支持(0) 反对(

#6楼[楼主] 2015-11-19 16:02 | shishanyuan

@ 张平a
不好意思，由于最近比较忙，不能及时回复
对于你遇到的问题个人觉得，一方面是spark与hive之间的版本支持，另一方面是spark现在还处于完善时期，可能比
现一些按照文档无法实现的问题，建议还是多试，多找一些资料参考

支持(0) 反对(

#7楼 2015-12-09 19:38 | appledx520

老师您好：我在运行sparksql时，一直卡在这个地方，没法解决，希望得到您的解答
15/12/09 19:30:44 INFO BlockManagerMasterActor: Registering block manager slaver6:38936 with 5
0.3 MB RAM
15/12/09 19:30:44 INFO BlockManagerMasterActor: Registering block manager slaver5:47349 with 5
0.3 MB RAM
15/12/09 19:30:44 INFO BlockManagerMasterActor: Registering block manager slaver1:47859 with 5
0.3 MB RAM
15/12/09 19:30:44 INFO BlockManagerMasterActor: Registering block manager slaver7:48437 with 5
0.3 MB RAM
15/12/09 19:30:44 INFO BlockManagerMasterActor: Registering block manager slaver3:49835 with 5
0.3 MB RAM
15/12/09 19:30:44 INFO BlockManagerMasterActor: Registering block manager slaver2:36605 with 5
0.3 MB RAM
15/12/09 19:30:44 INFO BlockManagerMasterActor: Registering block manager slaver4:36255 with 5
0.3 MB RAM

支持(0) 反对(

#8楼[楼主] 2015-12-10 12:14 | shishanyuan

@ appledx520
提供的信息是正常运行的日志，并没有异常的信息
按照你说的情况来看，是在注册Executor的BlockManager时卡住了，可以看一下是不是Master与Executor无法建
通讯（配置问题），还是存储空间不足等原因

支持(0) 反对(

#9楼 2015-12-10 12:18 | appledx520

谢谢您的回答，我的测试集群每台都是8g内存，应该不是内存不足。mysql和hive都配在主节点，master是可以ssl
到各个slaver的（slaver不能到master）

支持(0) 反对(

#10楼[楼主] 2015-12-10 14:49 | shishanyuan

@ appledx520
（slaver不能到master）--尽可能保证master与slaver之间互相ssh，确认几个问题：
0、hive是否安装成功，通过验证进行确认
1、在Spark的监控UI上，能不能看到Slave节点；
2、能不能看到启动的spark-sql应用程序，还没有启动成功，应该看不了:(
3、提供你的场景和其他日志信息

支持(0) 反对(

#11楼 2015-12-18 19:49 | zookeepers

老师，你好，我在eclipse里运行ide里的程序的时候出错了，不知道为什么，求解答

支持(0) 反对(

#12楼[楼主] 2015-12-18 21:15 | shishanyuan

@ zookeepers
通过你的截图可以看到"Required field 'client_protocol' is unset" ，应该是某个需要的字段没有初始化
检查一下Hive中的数据库是否正确初始化
另外你运行eclipse，可以通过提示的错误进行问题解决

支持(0) 反对(

#13楼 2016-04-13 09:33 | 唐予之

学习了，感谢！

支持(0) 反对(

#14楼 2016-07-13 14:47 | crazy_orange

写的太好了，感谢楼主！

支持(0) 反对(

#15楼[楼主] 2016-07-15 14:23 | shishanyuan

@ crazy_orange
谢谢！

支持(0) 反对(

#16楼 2016-07-22 09:29 | 飞飞_yang

博主老师，你好！我在SparkSQL操作的时候，比如我输入的指令是select count(*) from sogouq2;就会出现这样
问题；
16/07/22 09:25:14 WARN util.AkkaUtils: Error sending message in 1 attempts
akka.pattern.AskTimeoutException: Recipient[Actor[akka://sparkDriver/user/BlockManagerMaster#1
4126186]] had already been terminated.
at akka.pattern.AskableActorRef$.ask$extension(AskSupport.scala:134)
at org.apache.spark.util.AkkaUtils$.askWithReply(AkkaUtils.scala:175)
at org.apache.spark.storage.BlockManagerMaster.askDriverWithReply(BlockManagerMaster.scala:218
请问博主老师这是怎么问题导致的？谢谢你

支持(0) 反对(

#17楼[楼主] 2016-07-22 10:23 | shishanyuan

@ 飞飞_yang
从你的日志来看应该是通讯超时了：Recipient had already been terminated
你的操作是不是在操作Thrift Server出现的问题，检查一下配置是否正确？
特别检查所在节点Spark配置目录是否存在hive-site.xml文件

支持(0) 反对(

#18楼 2016-07-22 16:50 | 飞飞_yang

博主老师，您好！谢谢你的指点，我上午的问题解决了，上午是CLI命令操作，原因在于我只在master节点的conf下
置了hive-site.xml文件，没有在worker节点配置hive-site.xml文件。
下午我尝试了操作Thrift Server，我在jdbc.hive2://hadoop2:10000>操作数据的时候，当操作命令insert into
ble testThrift select ordernumber,amount from tbStockDetail where amount>3000;的时候，一直出现一
问题：
0: jdbc:hive2://hadoop2:10000> insert into table testThrift select c.theyear,max(d.sumofamount) fr
m tbDate c join (select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join
StockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber) d on c.dateid=d.
ateid group by c.theyear sort by c.theyear;
Error: java.lang.RuntimeException: org.apache.hadoop.security.AccessControlException: Permission
nied: user=hadoopadmin, access=EXECUTE, inode="/tmp":iespark:supergroup:drwx------
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkFsPermission(FSPermissionC
ecker.java:271)
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:
7)
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkTraverse(FSPermissionChec
r.java:208)
请问博主老师，这是什么原因导致的？谢谢老师

支持(0) 反对(

#19楼 2016-07-22 17:26 | 飞飞_yang

博主老师，您好！刚才那个问题，我已经解决了，我仔细看了一下报错日志，发现时hdf的权限问题，然后我就在ha
op下的hdfs-core.xml文件里修改了一下说明：
<property>
<name>dfs.permissions</name>
<value>false</value>
<description>
If "true", enable permission checking in HDFS.
If "false", permission checking is turned off,
but all other behavior is unchanged.
Switching from one parameter value to the other does not change the mode,
owner or group of files or directories.

</description>
</property>
然后再操作指令，就没问题了。谢谢博主老师！
祝博主老师工作顺利，身体健康！

支持(0) 反对(

#20楼 2016-07-23 14:55 | qqqqqhhhhh

老师您好，我想问一下，我在4.2.2节中show databases;始终没有找到hive数据库，请问是什么原因呢？

支持(0) 反对(

#21楼[楼主] 2016-07-26 15:57 | shishanyuan

@ qqqqqhhhhh
我也遇到同样的问题，当时我解决方法参考17楼、18楼的回复

支持(0) 反对(

#22楼 2016-08-15 11:55 | 飞飞_yang

老师，你好，我启动了spark-sql后，执行use hive;指令后，显示以下结果，然后就一直停留在那儿不动了，这是哪
出问题了，请老师给我指导下，谢谢老师！
16/08/15 11:52:00 INFO DAGScheduler: Job 6 finished: processCmd at CliDriver.java:376, took 0.06
421 s
sogouq1 false
sogouq2 false
tbdate false
tbstock false
tbstockdetail false
16/08/15 11:52:00 INFO CliDriver: Time taken: 0.092 seconds, Fetched 5 row(s)
spark-sql> 16/08/15 11:52:00 INFO StatsReportListener: Finished stage: org.apache.spark.schedule
StageInfo@7f943143
16/08/15 11:52:00 INFO StatsReportListener: task runtime:(count: 1, mean: 53.000000, stdev: 0.0
000, max: 53.000000, min: 53.000000)
16/08/15 11:52:00 INFO StatsReportListener: 0% 5% 10% 25% 50% 75% 90% 95% 100%
16/08/15 11:52:00 INFO StatsReportListener: 53.0 ms 53.0 ms 53.0 ms 53.0 ms 53.0 ms 53.0 ms 5
0 ms 53.0 ms 53.0 ms
16/08/15 11:52:00 INFO StatsReportListener: task result size:(count: 1, mean: 1143.000000, stdev
0.000000, max: 1143.000000, min: 1143.000000)
16/08/15 11:52:00 INFO StatsReportListener: 0% 5% 10% 25% 50% 75% 90% 95% 100%
16/08/15 11:52:00 INFO StatsReportListener: 1143.0 B 1143.0 B 1143.0 B 1143.0 B 1143.0 B 1143
B 1143.0 B 1143.0 B 1143.0 B
16/08/15 11:52:00 INFO StatsReportListener: executor (non-fetch) time pct: (count: 1, mean: 1.886
92, stdev: 0.000000, max: 1.886792, min: 1.886792)
16/08/15 11:52:00 INFO StatsReportListener: 0% 5% 10% 25% 50% 75% 90% 95% 100%
16/08/15 11:52:00 INFO StatsReportListener: 2 % 2 % 2 % 2 % 2 % 2 % 2 % 2 % 2 %
16/08/15 11:52:00 INFO StatsReportListener: other time pct: (count: 1, mean: 98.113208, stdev: 0
00000, max: 98.113208, min: 98.113208)
16/08/15 11:52:00 INFO StatsReportListener: 0% 5% 10% 25% 50% 75% 90% 95% 100%
16/08/15 11:52:00 INFO StatsReportListener: 98 % 98 % 98 % 98 % 98 % 98 % 98 % 98 % 98 %

支持(0) 反对(

#23楼 2016-08-23 19:29 | MatchYaya

"对于Spark1.1.0，对SQL表达式都作了CG优化，具体可以参看codegen模块。CG优化的实现主要还是依靠scala2
0的运行时放射机制（runtime reflection）" 这句话翻译的有问题吧 应该是反射机制 写成发生机制 产生歧义

支持(0) 反对(

#24楼 2017-02-27 15:08 | jyu_zxq

老师你好,spark-sql是否支持类似于hive的写法insert into table tab_name(field1,field2,...)values (....)或者是
nsert into table table_name(field1,field2,...) select field1,field2,... from other_table?我测试了一下,均提示
not support feature

支持(1) 反对(

#25楼[楼主] 2017-03-17 22:04 | shishanyuan

@ jyu_zxq
可以参见Hive的语法说明
https://cwiki.apache.org/confluence/display/Hive/LanguageManual

支持(0) 反对(

#26楼 2017-05-26 10:41 | 2017年1月18日

请问下，这些数据有建表语言吗？？？能提供下建表的语言吗，谢谢了

支持(0) 反对(

#27楼 2017-09-08 16:37 | peter.wei

spark sql 能支持 insert row 吗

我是指插入新的数据。

支持(0) 反对(

#28楼 2017-09-09 22:39 | 小狼崽一号

老师，我用hive.server2.thrift.bind.host默认的本地配置，连接sparksql后能成功查询到hive的数据，可是绑定到它服务器，能连接成功，但是查询不到hive的数据，是什么原因呢？求解，谢谢。。

```
0: jdbc:hive2://hadoop08:10000> show databases;
+---------------+--+
| databaseName  |
+---------------+--+
| default       |
+---------------+--+
```

支持(0) 反对(

刷新评论 刷新页面 返回顶

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。