SparkSQL作业

目录:

题目1

为Spark SQL添加一条自定义命令

• SHOW VERSION
• 显示当前Spark版本和Java版本

Spark源码编译问题

问题1

[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:3.0.0-M1:test (default-test) on project leshan-integration-tests: There are test failures.

解决方法

使用插件，在相应pom.xml文件里面，修改

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <configuration>
                <testFailureIgnore>true</testFailureIgnore>
            </configuration>
        </plugin>
    </plugins>
 </build>
```

问题2

Failed to execute goal org.apache.maven.plugins:maven-antrun-plugin:1.8:run (default) on project spark-core_2.12: An Ant BuildException has occured: Execute failed: java.io.IOException: Cannot run program "bash" : CreateProcess error=2

解决办法

在spark源码目录下使用git bash执行命令

```
mvn clean package -DskipTests -Phive -Phive-thriftserver
```

Maven 编译

```
mvn clean package -DskipTests -Phive -Phive-thriftserver
```

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Spark Project Parent POM 3.2.0-SNAPSHOT:
[INFO]
[INFO] Spark Project Parent POM ........................... SUCCESS [  3.274 s]
[INFO] Spark Project Tags ................................. SUCCESS [  5.364 s]
[INFO] Spark Project Sketch ............................... SUCCESS [  6.136 s]
[INFO] Spark Project Local DB ............................. SUCCESS [  2.971 s]
[INFO] Spark Project Networking ........................... SUCCESS [  5.393 s]
[INFO] Spark Project Shuffle Streaming Service ............ SUCCESS [  3.618 s]
[INFO] Spark Project Unsafe ............................... SUCCESS [  8.219 s]
[INFO] Spark Project Launcher ............................. SUCCESS [  2.544 s]
[INFO] Spark Project Core ................................. SUCCESS [03:17 min]
[INFO] Spark Project ML Local Library ..................... SUCCESS [ 32.745 s]
[INFO] Spark Project GraphX ............................... SUCCESS [ 32.637 s]
[INFO] Spark Project Streaming ............................ SUCCESS [01:44 min]
[INFO] Spark Project Catalyst ............................. SUCCESS [05:52 min]
[INFO] Spark Project SQL .................................. SUCCESS [09:09 min]
[INFO] Spark Project ML Library ........................... SUCCESS [04:24 min]
[INFO] Spark Project Tools ................................ SUCCESS [  7.317 s]
[INFO] Spark Project Hive ................................. SUCCESS [02:13 min]
[INFO] Spark Project REPL ................................. SUCCESS [ 16.415 s]
[INFO] Spark Project Hive Thrift Server ................... SUCCESS [ 55.491 s]
[INFO] Spark Project Assembly ............................. SUCCESS [  4.895 s]
[INFO] Kafka 0.10+ Token Provider for Streaming ........... SUCCESS [ 18.291 s]
[INFO] Spark Integration for Kafka 0.10 ................... SUCCESS [ 24.881 s]
[INFO] Kafka 0.10+ Source for Structured Streaming ........ SUCCESS [ 41.144 s]
[INFO] Spark Project Examples ............................. SUCCESS [ 47.416 s]
[INFO] Spark Integration for Kafka 0.10 Assembly .......... SUCCESS [ 12.820 s]
[INFO] Spark Avro ......................................... SUCCESS [ 37.400 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  32:51 min
[INFO] Finished at: 2021-09-04T18:23:30+08:00
[INFO] ------------------------------------------------------------------------
```

```
[INFO] --- maven-source-plugin:3.1.0:test-jar-no-fork (create-source-jar) @ spark-avro_2.12 ---
[INFO] Building jar: D:\sourcecode\sparksql\spark\external\avro\target\spark-avro_2.12-3.3.0-SNAPSHOT-test-sources.jar
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Spark Project Parent POM 3.3.0-SNAPSHOT:
[INFO]
[INFO] Spark Project Parent POM ........................... SUCCESS [  3.638 s]
[INFO] Spark Project Tags ................................. SUCCESS [ 16.894 s]
[INFO] Spark Project Sketch ............................... SUCCESS [ 10.734 s]
[INFO] Spark Project Local DB ............................. SUCCESS [  3.218 s]
[INFO] Spark Project Networking ........................... SUCCESS [  7.696 s]
[INFO] Spark Project Shuffle Streaming Service ............ SUCCESS [  5.061 s]
[INFO] Spark Project Unsafe ............................... SUCCESS [ 10.056 s]
[INFO] Spark Project Launcher ............................. SUCCESS [  1.698 s]
[INFO] Spark Project Core ................................. SUCCESS [02:28 min]
[INFO] Spark Project ML Local Library ..................... SUCCESS [ 37.365 s]
[INFO] Spark Project GraphX ............................... SUCCESS [ 30.036 s]
[INFO] Spark Project Streaming ............................ SUCCESS [01:00 min]
[INFO] Spark Project Catalyst ............................. SUCCESS [02:40 min]
[INFO] Spark Project SQL .................................. SUCCESS [03:21 min]
[INFO] Spark Project ML Library ........................... SUCCESS [02:13 min]
[INFO] Spark Project Tools ................................ SUCCESS [  4.920 s]
[INFO] Spark Project Hive ................................. SUCCESS [01:26 min]
[INFO] Spark Project REPL ................................. SUCCESS [ 19.510 s]
[INFO] Spark Project Hive Thrift Server ................... SUCCESS [ 44.565 s]
[INFO] Spark Project Assembly ............................. SUCCESS [  6.737 s]
[INFO] Kafka 0.10+ Token Provider for Streaming ........... SUCCESS [ 16.629 s]
[INFO] Spark Integration for Kafka 0.10 ................... SUCCESS [ 21.259 s]
[INFO] Kafka 0.10+ Source for Structured Streaming ........ SUCCESS [ 46.235 s]
[INFO] Spark Project Examples ............................. SUCCESS [ 36.745 s]
[INFO] Spark Integration for Kafka 0.10 Assembly .......... SUCCESS [  8.303 s]
[INFO] Spark Avro ......................................... SUCCESS [ 34.345 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  19:16 min
[INFO] Finished at: 2021-09-05T12:26:21+08:00
[INFO] ------------------------------------------------------------------------
```

结果验证

Spark-sql 启动

windows 下编译成功了，启动spark-sql 验证结果时一直报错，解决不了

```
$ bin/spark-sql.cmd -S
Exception in thread "main" java.lang.UnsatisfiedLinkError:
org.apache.hadoop.io.nativeio.NativeIO$POSIX.stat(Ljava/lang/String;)Lorg/apache
/hadoop/io/nativeio/NativeIO$POSIX$Stat;
        at org.apache.hadoop.io.nativeio.NativeIO$POSIX.stat(Native Method)
        at
org.apache.hadoop.io.nativeio.NativeIO$POSIX.getStat(NativeIO.java:460)
```

```
        at
org.apache.hadoop.fs.RawLocalFileSystem$DeprecatedRawLocalFileStatus.loadPermiss
ionInfoByNativeIO(RawLocalFileSystem.java:821)
        at
org.apache.hadoop.fs.RawLocalFileSystem$DeprecatedRawLocalFileStatus.loadPermiss
ionInfo(RawLocalFileSystem.java:735)
        at
org.apache.hadoop.fs.RawLocalFileSystem$DeprecatedRawLocalFileStatus.getPermissi
on(RawLocalFileSystem.java:703)
        at
org.apache.hadoop.hive.ql.session.SessionState.createRootHDFSDir(SessionState.ja
va:711)
        at
org.apache.hadoop.hive.ql.session.SessionState.createSessionDirs(SessionState.ja
va:654)
        at
org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:586)
        at
org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:548)
        at
org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver$.main(SparkSQLCLIDriver
.scala:135)
        at
org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver.main(SparkSQLCLIDriver.
scala)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.jav
a:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at
org.apache.spark.deploy.JavaMainApplication.start(SparkApplication.scala:52)
        at
org.apache.spark.deploy.SparkSubmit.org$apache$spark$deploy$SparkSubmit$$runMain
(SparkSubmit.scala:928)
        at
org.apache.spark.deploy.SparkSubmit.doRunMain$1(SparkSubmit.scala:180)
        at org.apache.spark.deploy.SparkSubmit.submit(SparkSubmit.scala:203)
        at org.apache.spark.deploy.SparkSubmit.doSubmit(SparkSubmit.scala:90)
        at
org.apache.spark.deploy.SparkSubmit$$anon$2.doSubmit(SparkSubmit.scala:1007)
        at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:1016)
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

Building a Runnable Distribution

build 成功后， windows下spark-sql启动不了，按照官网build spark 思路， 打成tgz包，发布到linux 环境验证

```
# 打包命令帮助
./dev/make-distribution.sh --help

# Building a Runnable Distribution
./dev/make-distribution.sh --name spark-3.2  --tgz -Phive -Phive-thriftserver  -
Pyarn -DskipTests  -Dmaven.compile.fork=true -T 1C clean package

# 如果需要是堆内存不够，则设置最大堆大小-Xmx，如果是持久代溢出，比如出现PermGen space异常，则
设置-XX:MaxPermSize即可
 set MAVEN_OPTS= -Xms800m -Xmx800m -XX:MaxNewSize=512m  -XX:MaxPermSize=512m
export MAVEN_OPTS="-Xms6g -Xmx6g -XX:+UseG1GC -XX:ReservedCodeCacheSize=2g"

增加跳过测试代码的编译命令：  -Dmaven.test.skip=true
指明多线程进行编译：-Dmaven.compile.fork=true
增加CPU核心参数：-T 1C
```

./dev/make-distribution.sh --name spark-3.2 --tgz -Phive -Phive-thriftserver -Pyarn -DskipTests -Dmaven.compile.fork=true -T 1C clean package windows下执行脚本一直报错解决不掉。

上传到Linux集群

一直报错，因为windwos下是\r\n换行， linux 是\n ，这个脚本我都从dos改成unix ， 可能还是格式不对， 多了一个\r ，

把其他可以运行的spark sql 拷贝过来一个，依然报这个错

题目2

构建SQL满足如下要求

通过set spark.sql.planChangeLog.level=WARN;查看

1. 构建一条SQL，同时apply下面三条优化规则：
   CombineFilters
   CollapseProject
   BooleanSimplification
2. 构建一条SQL，同时apply下面五条优化规则：
   ConstantFolding
   PushDownPredicates
   ReplaceDistinctWithAggregate
   ReplaceExceptWithAntiJoin
   FoldablePropagation

准备工作

```
# 建表语句
create table tmp (id int , name String);
# 插入数据
insert into tmp values(1,'wanghuan');
insert into tmp values(1,'fanfan');
insert into tmp values(1,'doudou');
insert into tmp values(1,'tom');


set spark.sql.planChangeLog.level=WARN;
```

1. 第一条SQL

```
select name from (
  select id+(1+2) as id, name from tmp a where id =1 and 1 = 1
) where name='wanghuan';
```

planChangeLog

采用PushDownPredicates 替代了CombineFilters 规则， CombineFilters 算子没有写出对应SQL

```
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.PushDownPredicates ===
 Project [name#51]
                Project [name#51]
!+- Filter (name#51 = wanghuan)
                   +- Project [(id#50 + (1 + 2)) AS id#49, name#51]
!   +- Project [(id#50 + (1 + 2)) AS id#49, name#51]
                      +- Filter (((id#50 = 1) AND (1 = 1)) AND (name#51 = wanghuan))
!      +- Filter ((id#50 = 1) AND (1 = 1))
                         +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, na
me#51], Partition Cols: []]
!         +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, name#51], Parti
tion Cols: []]
```

```
21/09/05 15:06:57 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.CollapseProject ===
 Project [name#51]
             Project [name#51]
!+- Project [name#51]
                +- Filter (((id#50 = 1) AND (1 = 1)) AND (name#51 = wanghuan))
!   +- Filter (((id#50 = 1) AND (1 = 1)) AND (name#51 = wanghuan))
                   +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, name#51]
, Partition Cols: []]
!      +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, name#51], Partitio
n Cols: []]

21/09/05 15:06:57 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.ConstantFolding ===
 Project [name#51]
             Project [name#51]
!+- Filter (((id#50 = 1) AND (1 = 1)) AND (name#51 = wanghuan))
             +- Filter (((id#50 = 1) AND true) AND (name#51 = wanghuan))
    +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, name#51], Partition C
ols: []]      +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, name#51], P
artition Cols: []]

21/09/05 15:06:57 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.BooleanSimplification ===
 Project [name#51]
             Project [name#51]
!+- Filter (((id#50 = 1) AND true) AND (name#51 = wanghuan))
             +- Filter ((id#50 = 1) AND (name#51 = wanghuan))
    +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, name#51], Partition C
ols: []]      +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#50, name#51], P
artition Cols: []]
```

2.第二条SQL

```
select distinct name , 'MAX' AS A from (
  select id+(1+2) as id, name from tmp a where id IN (1,2,3) and 1 = 1
  except select id+(1+2) as id, name from tmp a where id =1
) where name='wanghuan'
order by A desc ;
```

planChangeLog

```
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.ReplaceDistinctWithAggregate ===
!Distinct                                                 Aggregate [name#55], [name#55]
 +- Project [name#55]                                     +- Project [name#55]
    +- Filter (name#55 = wanghuan)                           +- Filter (name#55 = wanghuan)
       +- Project [(id#54 + (1 + 2)) AS id#53, name#55]         +- Project [(id#54 + (1 + 2)) AS id#53, name#55]
          +- Filter ((id#54 = 1) AND (1 = 1))                      +- Filter ((id#54 = 1) AND (1 = 1))
             +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#54, name#55], Partition Cols: []]              +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#54, name#55], Partition Cols: []]
```

```
21/09/05 15:11:09 WARN PlanChangeLogger: Batch Aggregate has no effect.
21/09/05 15:11:09 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.PushDownPredicates ===
 Aggregate [name#55], [name#55]                           Aggregate [name#55], [name#55]
 +- Project [name#55]                                     +- Project [name#55]
!   +- Filter (name#55 = wanghuan)                           +- Project [(id#54 + (1 + 2)) AS id#53, name#55]
!      +- Project [(id#54 + (1 + 2)) AS id#53, name#55]         +- Filter (((id#54 = 1) AND (1 = 1)) AND (name#55 = wanghuan))
!         +- Filter ((id#54 = 1) AND (1 = 1))                      +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#54, name#55], Partition Cols: []]
!            +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#54, name#55], Partition Cols: []]
```

```
21/09/05 15:11:09 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.ConstantFolding ===
 Aggregate [name#55], [name#55]                           Aggregate [name#55], [name#55]
 +- Project [name#55]                                     +- Project [name#55]
!   +- Filter (((id#54 = 1) AND (1 = 1)) AND (name#55 = wanghuan))   +- Filter (((id#54 = 1) AND true) AND (name#55 = wanghuan))
       +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#54, name#55], Partition Cols: []]        +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#54, name#55], Partition Cols: []]
```

```
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.ReplaceExceptWithFilter ===
 Distinct                                                 Distinct
 +- Project [name#42]                                     +- Project [name#42]
    +- Filter (name#42 = wanghuan)                           +- Filter (name#42 = wanghuan)
!      +- Except false                                          +- Distinct
!         :- Project [(id#41 + (1 + 2)) AS id#39, name#42]         +- Filter NOT coalesce((id#39 = 1), false)
!         :  +- Filter (id#41 IN (1,2,3) AND (1 = 1))                 +- Project [(id#41 + (1 + 2)) AS id#39, name#42]
!         :     +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#41, name#42], Partition Cols: []]                  +- Filter (id#41 IN (1,2,3) AND (1 = 1))
!         +- Project [(id#43 + (1 + 2)) AS id#40, name#44]            +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#41, name#42], Partition Cols: []]
!            +- Filter (id#43 = 1)
!               +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#43, name#44], Partition Cols: []]
```

```
21/09/05 15:28:36 WARN PlanChangeLogger:
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.FoldablePropagation ===
!Sort [A#48 DESC NULLS LAST], true                        Sort [MAX DESC NULLS LAST], true
!+- Aggregate [name#50, A#48], [name#50, A#48]              +- Aggregate [name#50, MAX], [name#50, MAX AS A#48]
    +- Aggregate [id#46, name#50], [name#50, MAX AS A#48]       +- Aggregate [id#46, name#50], [name#50, MAX AS A#48]
       +- Filter (NOT coalesce((id#46 = 1), false) AND (name#50 = wanghuan))        +- Filter (NOT coalesce((id#46 = 1), false) AND (name#50 = wanghuan))
          +- Project [(id#49 + (1 + 2)) AS id#46, name#50]             +- Project [(id#49 + (1 + 2)) AS id#46, name#50]
             +- Filter (id#49 IN (1,2,3) AND (1 = 1))                     +- Filter (id#49 IN (1,2,3) AND (1 = 1))
                +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#49, name#50], Partition Cols: []]                 +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#49, name#50], Partition Cols: []]
```

题目3

实现自定义优化规则（静默规则）

第一步实现自定义规则（静默规则，通过set spark.sql.planChangeLog.level=WARN;确认执行到就行）
case class MyPushDown(spark: SparkSession) extends Rule[LogicalPlan] {
def apply(plan: LogicalPlan): LogicalPlan = plan transform { .... }
}
第二步创建自己的Extension并注入
class MySparkSessionExtension extends (SparkSessionExtensions => Unit) {

override def apply(extensions: SparkSessionExtensions): Unit = {
extensions.injectOptimizerRule { session =>
new MyPushDown(session)
}
}
}
第三步通过spark.sql.extensions提交
bin/spark-sql --jars /opt/sourcecode/CustomSparkSessionExtension-1.0-SNAPSHOT.jar --conf spark.sql.extensions=org.example.MyCustomSparkExtension

POM

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>CustomSparkSessionExtension</artifactId>
    <version>1.0-SNAPSHOT</version>
    <inceptionYear>2021</inceptionYear>
    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <scala.version>2.12</scala.version>
        <scala.binary.version>2.12.10</scala.binary.version>
        <spark.version>3.1.2</spark.version>
    </properties>

    <repositories>
        <repository>
            <id>maven-ali</id>
            <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_${scala.version}</artifactId>
            <version>${spark.version}</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-sql_${scala.version}</artifactId>
            <version>${spark.version}</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-catalyst_${scala.version}</artifactId>
            <version>${spark.version}</version>
            <scope>provided</scope>
        </dependency>
```

```
    </dependencies>

    <build>
        <sourceDirectory>src/main/scala</sourceDirectory>
        <testSourceDirectory>src/test/scala</testSourceDirectory>
        <plugins>
            <!-- 该插件将scala代码编译成class文件 -->
            <plugin>
                <groupId>net.alchim31.maven</groupId>
                <artifactId>scala-maven-plugin</artifactId>
                <version>4.3.0</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

MyPushDown

实现自定义规则

```
package org.example

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.catalyst.plans.logical.LogicalPlan
import org.apache.spark.sql.catalyst.rules.Rule
import org.apache.spark.sql.catalyst.expressions.{Literal, Multiply}
import org.apache.spark.sql.types.Decimal

case class MyPushDown(spark: SparkSession) extends Rule[LogicalPlan] {
  override def apply(plan: LogicalPlan): LogicalPlan = {
    println("开始应用 MyPushDown 优化规则")
    plan transformAllExpressions {
      case Multiply(left, right,true) if right.isInstanceOf[Literal] &&
        right.asInstanceOf[Literal].value.isInstanceOf[Decimal] &&
        right.asInstanceOf[Literal].value.asInstanceOf[Decimal].toDouble == 1.0
=>
        println("MyPushDown 优化规则生效")
        left
    }
  }
}
```

MyCustomSparkExtension

创建自己的Extension并注入

```
package org.example

import org.apache.spark.sql.SparkSessionExtensions
import org.apache.spark.sql.catalyst.plans.logical.LogicalPlan
import org.apache.spark.sql.catalyst.rules.Rule
import org.apache.spark.sql.SparkSession

class MyCustomSparkExtension extends (SparkSessionExtensions => Unit) {
  override def apply(extensions: SparkSessionExtensions): Unit = {
    extensions.injectOptimizerRule { session =>
      new MyPushDown(session)
    }
  }
}
```

spark.sql.extensions提交

```
bin/spark-sql --jars /opt/sourcecode/CustomSparkSessionExtension-1.0-
SNAPSHOT.jar --conf spark.sql.extensions=org.example.MyCustomSparkExtension

方法1 ：
spark-sql> explain extended select id * 1.0 from tmp ;

方法2 ：
spark-sql> set spark.sql.planChangeLog.level=WARN;
spark-sql> explain extended select id * 1.0 from tmp ;
```

验证结果

```
=== Applying Rule org.apache.spark.sql.catalyst.optimizer.ConstantFolding ===
!Project [CheckOverflow((promote_precision(cast(cast(id#34 as decimal(10,0)) as decimal(11,1))) * promote_precision(cast(1.0 as decimal(11,
1)))), DecimalType(13,1), true) AS (CAST(CAST(id AS DECIMAL(10,0)) AS DECIMAL(11,1)) * CAST(1.0 AS DECIMAL(11,1)))#36]   Project [CheckOver
flow((promote_precision(cast(cast(id#34 as decimal(10,0)) as decimal(11,1))) * 1.0), DecimalType(13,1), true) AS (CAST(CAST(id AS DECIMAL(1
0,0)) AS DECIMAL(11,1)) * CAST(1.0 AS DECIMAL(11,1)))#36]
 +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#34, name#35], Partition Cols: []
]                                                                                                                   +- HiveTableRelati
on [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#34, name#35], Partition Cols: []]

开始应用 MyPushDown 优化规则
开始应用 MyPushDown 优化规则
21/09/05 17:12:46 WARN PlanChangeLogger:
=== Result of Batch Operator Optimization before Inferring Filters ===
!Project [CheckOverflow((promote_precision(cast(cast(id#34 as decimal(10,0)) as decimal(11,1))) * promote_precision(cast(1.0 as decimal(11,
1)))), DecimalType(13,1), true) AS (CAST(CAST(id AS DECIMAL(10,0)) AS DECIMAL(11,1)) * CAST(1.0 AS DECIMAL(11,1)))#36]   Project [CheckOver
flow((promote_precision(cast(cast(id#34 as decimal(10,0)) as decimal(11,1))) * 1.0), DecimalType(13,1), true) AS (CAST(CAST(id AS DECIMAL(1
0,0)) AS DECIMAL(11,1)) * CAST(1.0 AS DECIMAL(11,1)))#36]
 +- HiveTableRelation [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#34, name#35], Partition Cols: []
]                                                                                                                   +- HiveTableRelati
on [`default`.`tmp`, org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, Data Cols: [id#34, name#35], Partition Cols: []]

21/09/05 17:12:46 WARN PlanChangeLogger: Batch Infer Filters has no effect.
开始应用 MyPushDown 优化规则
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Operator Optimization after Inferring Filters has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Push extra predicate through join has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Early Filter and Projection Push-Down has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Join Reorder has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Eliminate Sorts has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Decimal Optimizations has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Distinct Aggregate Rewrite has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Object Expressions Optimization has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch LocalRelation has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch Check Cartesian Products has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch RewriteSubquery has no effect.
21/09/05 17:12:46 WARN PlanChangeLogger: Batch NormalizeFloatingNumbers has no effect.
```