Debugging Planning Issues Using Calcite's Built-in Loggers

Stamatis Zampetakis
Alessandro Solimando

APACHE calcite™

Virtual Meetup
March 15, 2023

About us

Alessandro Solimando @asolimando
Senior Software Engineer @ Rakuten*
Committer of Apache Calcite & Apache Hive
PhD in Data Management, University of Genoa
(* the work presented here was done while at Cloudera)

Stamatis Zampetakis @szampetak
Staff Software Engineer @ Cloudera, Hive query optimizer team
PMC member of Apache Calcite & Apache Hive
PhD in Data Management, INRIA & Paris-Sud University

Outline

- Motivation / Common planning issues
- Calcite’s built-in logging (RuleEventLogger)
  - Configuration (XML/Properties)
  - Output explained
- Hive case studies:
  - Hanging TPC-DS queries
  - OutOfMemoryError (HIVE-25758)
  - Wrong results (HIVE-26722)
- Conclusion

Common Planning Issues

Issue	**Diagnostic Information**
OutOfMemoryError	Heap dumps
StackOverflowError	Stack traces
Unresponsive server	Stack traces
Very slow (or hanging) queries	Stack traces
Wrong results	Query plans
Query crashes (NPE, AssertionError, ClassCastException)	Stack traces

Common Planning Issues

Issue	Diagnostic Information
OutOfMemoryError	Heap dumps
StackOverflowError	Stack traces
Unresponsive server	Stack traces
Very slow (or hanging) queries	Stack traces
Wrong results	Query plans
Query crashes (NPE, AssertionError, ClassCastException)	Stack traces

- Many times the **root cause** lies in **rule based** transformations

Common Planning Issues

Issue	**Diagnostic Information**
OutOfMemoryError	Heap dumps
StackOverflowError	Stack traces
Unresponsive server	Stack traces
Very slow (or hanging) queries	Stack traces
Wrong results	Query plans
Query crashes (NPE, AssertionError, ClassCastException)	Stack traces

- Many times the **root cause** lies in **rule based** transformations
- Good **logs** are **key** for finding this **root cause**

Common Planning Issues

- Interactive debugging not always feasible:
  - CI/CU environment with restricted access
  - Data specific problem
- Debugging planning issues interactively (without logs) is difficult:
  - Large number of transformations
  - Complex transformation logic
  - Lots of intermediate objects
  - Repeated occurrences of transformations

Built-in logging: RuleEventLogger

RuleEventLogger - XML Configuration

```
<Configuration>
 <Appenders>
   <Console name="A1" target="SYSTEM_OUT">
     <PatternLayout
         pattern="%m%n"/>
   </Console>
 </Appenders>
 <Loggers>
   <Root level="INFO">
     <AppenderRef ref="A1"/>
   </Root>
   <logger name="org.apache.calcite.plan.RelOptPlanner"level="DEBUG">
     <MarkerFilter marker="FULL_PLAN" onMatch="DENY" onMismatch="NEUTRAL"/>
   </logger>
 </Loggers>
</Configuration>
```

https://logging.apache.org/log4j/2.x/manual/configuration.html#automatic-configuration

Calcite unit tests: Add/Modify core/src/test/resources/**log4j2-test.xml** file

RuleEventLogger - XML Configuration

```
<Configuration>
 <Appenders>
   <Console name="A1" target="SYSTEM_OUT">
     <PatternLayout
         pattern="%m%n"/>
   </Console>
 </Appenders>
 <Loggers>
   <Root level="INFO">
     <AppenderRef ref="A1"/>
   </Root>
   <logger name="org.apache.calcite.plan.RelOptPlanner"level="DEBUG">
     <MarkerFilter marker="FULL_PLAN" onMatch="DENY" onMismatch="NEUTRAL"/>
   </logger>
 </Loggers>
</Configuration>
```

https://logging.apache.org/log4j/2.x/manual/configuration.html#automatic-configuration

Calcite unit tests: Add/Modify core/src/test/resources/**log4j2-test.xml** file

RuleEventLogger - XML Configuration

```
<Configuration>
 <Appenders>
   <Console name="A1" target="SYSTEM_OUT">
     <PatternLayout
         pattern="%m%n"/>
   </Console>
 </Appenders>
 <Loggers>
   <Root level="INFO">
     <AppenderRef ref="A1"/>
   </Root>
   <logger name="org.apache.calcite.plan.RelOptPlanner"level="DEBUG">
     <MarkerFilter marker="FULL_PLAN" onMatch="DENY" onMismatch="NEUTRAL"/>
   </logger>
 </Loggers>
</Configuration>
```

CALCITE-4704 (1.29.0)

CALCITE-4991 (1.30.0)

https://logging.apache.org/log4j/2.x/manual/configuration.html#automatic-configuration

Calcite unit tests: Add/Modify core/src/test/resources/**log4j2-test.xml** file

RuleEventLogger - Properties Configuration

```
appenders = A1

appender.A1.type = Console
appender.A1.name = A1
appender.A1.target = SYSTEM_OUT
appender.A1.layout.type = PatternLayout
appender.A1.layout.pattern = %m%n

loggers = CBORuleLogger

rootLogger.level = INFO
rootLogger.appenderRefs = A1
rootLogger.appenderRef.A1.ref = A1

logger.CBORuleLogger.name = org.apache.calcite.plan.RelOptPlanner
logger.CBORuleLogger.level = DEBUG
logger.CBORuleLogger.filter.marker.type= MarkerFilter
logger.CBORuleLogger.filter.marker.marker= FULL_PLAN
logger.CBORuleLogger.filter.marker.onMatch= DENY
logger.CBORuleLogger.filter.marker.onMismatch= NEUTRAL
```

CALCITE-4704 (1.29.0)

CALCITE-4991 (1.30.0)

https://logging.apache.org/log4j/2.x/manual/configuration.html#automatic-configuration

Calcite unit tests: Add/Modify core/src/test/resources/**log4j2-test.properties** file

RuleEventLogger - Properties Configuration (HIVE)

```
appenders = A1

appender.A1.type = Console
appender.A1.name = A1
appender.A1.target = SYSTEM_OUT
appender.A1.layout.type = PatternLayout
appender.A1.layout.pattern = %m%n

loggers = …,CBORuleLogger

rootLogger.level = INFO
rootLogger.appenderRefs = A1
rootLogger.appenderRef.A1.ref = A1

logger.CBORuleLogger.name = org.apache.hadoop.hive.ql.optimizer.calcite.RuleEventLogger
logger.CBORuleLogger.level = DEBUG
logger.CBORuleLogger.filter.marker.type= MarkerFilter
logger.CBORuleLogger.filter.marker.marker= FULL_PLAN
logger.CBORuleLogger.filter.marker.onMatch= DENY
logger.CBORuleLogger.filter.marker.onMismatch= NEUTRAL
```

HIVE-25816

Hive tests: Modify data/conf/**hive-log4j2.properties** file

RuleEventLogger - Example

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= DENY
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= DENY
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#0: Apply rule [AggregateRemoveRule] to [rel#11:LogicalAggregate]
    call#0: Rule [AggregateRemoveRule] produced [rel#15:LogicalProject]
    call#1: Apply rule [ProjectMergeRule] to [rel#13:LogicalProject,rel#15:LogicalProject]
    call#1: Rule [ProjectMergeRule] produced [rel#17:LogicalProject]
    call#2: Apply rule [ProjectMergeRule] to [rel#17:LogicalProject,rel#9:LogicalProject]
    call#2: Rule [ProjectMergeRule] produced [rel#19:LogicalProject]
```

RuleEventLogger - Output Explained

```sql
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= DENY
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#0: Apply rule [AggregateRemoveRule] to [rel#11:LogicalAggregate]
    call#0: Rule [AggregateRemoveRule] produced [rel#15:LogicalProject]
    call#1: Apply rule [ProjectMergeRule] to [rel#13:LogicalProject,rel#15:LogicalProject]
    call#1: Rule [ProjectMergeRule] produced [rel#17:LogicalProject]
    call#2: Apply rule [ProjectMergeRule] to [rel#17:LogicalProject,rel#9:LogicalProject]
    call#2: Rule [ProjectMergeRule] produced [rel#19:LogicalProject]
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= DENY
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#0: Apply rule [AggregateRemoveRule] to [rel#11:LogicalAggregate]
    call#0: Rule [AggregateRemoveRule] produced [rel#15:LogicalProject]
    call#1: Apply rule [ProjectMergeRule] to [rel#13:LogicalProject,rel#15:LogicalProject]
    call#1: Rule [ProjectMergeRule] produced [rel#17:LogicalProject]
    call#2: Apply rule [ProjectMergeRule] to [rel#17:LogicalProject,rel#9:LogicalProject]
    call#2: Rule [ProjectMergeRule] produced [rel#19:LogicalProject]
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= DENY
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#0: Apply rule [AggregateRemoveRule] to [rel#11:LogicalAggregate]
    call#0: Rule [AggregateRemoveRule] produced [rel#15:LogicalProject]
    call#1: Apply rule [ProjectMergeRule] to [rel#13:LogicalProject,rel#15:LogicalProject]
    call#1: Rule [ProjectMergeRule] produced [rel#17:LogicalProject]
    call#2: Apply rule [ProjectMergeRule] to [rel#17:LogicalProject,rel#9:LogicalProject]
    call#2: Rule [ProjectMergeRule] produced [rel#19:LogicalProject]
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= DENY
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#0: Apply rule [AggregateRemoveRule] to [rel#11:LogicalAggregate]
    call#0: Rule [AggregateRemoveRule] produced [rel#15:LogicalProject]
    call#1: Apply rule [ProjectMergeRule] to [rel#13:LogicalProject,rel#15:LogicalProject]
    call#1: Rule [ProjectMergeRule] produced [rel#17:LogicalProject]
    call#2: Apply rule [ProjectMergeRule] to [rel#17:LogicalProject,rel#9:LogicalProject]
    call#2: Rule [ProjectMergeRule] produced [rel#19:LogicalProject]
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= ACCEPT
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#0: Apply rule [AggregateRemoveRule] to [rel#11:LogicalAggregate]
    call#0: Full plan for rule input [rel#11:LogicalAggregate]:
    LogicalAggregate(group=[{0, 1}], EXPR$1=[COUNT($2)])
      LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
        LogicalTableScan(table=[[CATALOG, SALES, EMP]])

    call#0: Rule [AggregateRemoveRule] produced [rel#15:LogicalProject]
    call#0: Full plan for [rel#15:LogicalProject]:
    LogicalProject(EMPNO=[$0], DEPTNO=[$1], $f2=[CASE(IS NOT NULL($2), 1:BIGINT, 0:BIGINT)])
      LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
        LogicalTableScan(table=[[CATALOG, SALES, EMP]])
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= ACCEPT
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#0: Apply rule [AggregateRemoveRule] to [rel#11:LogicalAggregate]
    call#0: Full plan for rule input [rel#11:LogicalAggregate]:
    LogicalAggregate(group=[{0, 1}], EXPR$1=[COUNT($2)])
      LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
        LogicalTableScan(table=[[CATALOG, SALES, EMP]])

    call#0: Rule [AggregateRemoveRule] produced [rel#15:LogicalProject]
    call#0: Full plan for [rel#15:LogicalProject]:
    LogicalProject(EMPNO=[$0], DEPTNO=[$1], $f2=[CASE(IS NOT NULL($2), 1:BIGINT, 0:BIGINT)])
      LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
        LogicalTableScan(table=[[CATALOG, SALES, EMP]])
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= ACCEPT
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#1: Apply rule [ProjectMergeRule] to [rel#13:LogicalProject,rel#15:LogicalProject]
    call#1: Full plan for rule input [rel#13:LogicalProject]:
    LogicalProject(EMPNO=[$0], EXPR$1=[$2])
      LogicalProject(EMPNO=[$0], DEPTNO=[$1], $f2=[CASE(IS NOT NULL($2), 1:BIGINT, 0:BIGINT)])
        LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
          LogicalTableScan(table=[[CATALOG, SALES, EMP]])

    call#1: Full plan for rule input [rel#15:LogicalProject]:
    ...

    call#1: Rule [ProjectMergeRule] produced [rel#17:LogicalProject]
    call#1: Full plan for [rel#17:LogicalProject]:
    LogicalProject(EMPNO=[$0], EXPR$1=[CASE(IS NOT NULL($2), 1:BIGINT, 0:BIGINT)])
      LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
        LogicalTableScan(table=[[CATALOG, SALES, EMP]])
```

RuleEventLogger - Output Explained

```
SELECT empno, count(mgr) FROM sales.emp GROUP BY empno, deptno
```

```
logger.CBORuleLogger.filter.marker.onMatch= ACCEPT
```

```
./gradlew :core:test --tests RelOptRulesTest.testAggregateRemove3

RelOptRulesTest > testAggregateRemove3() STANDARD_OUT
    call#2: Apply rule [ProjectMergeRule] to [rel#17:LogicalProject,rel#9:LogicalProject]
    call#2: Full plan for rule input [rel#17:LogicalProject]:
    LogicalProject(EMPNO=[$0], EXPR$1=[CASE(IS NOT NULL($2), 1:BIGINT, 0:BIGINT)])
      LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
        LogicalTableScan(table=[[CATALOG, SALES, EMP]])

    call#2: Full plan for rule input [rel#9:LogicalProject]:
    LogicalProject(EMPNO=[$0], DEPTNO=[$7], MGR=[$3])
      LogicalTableScan(table=[[CATALOG, SALES, EMP]])

    call#2: Rule [ProjectMergeRule] produced [rel#19:LogicalProject]
    call#2: Full plan for [rel#19:LogicalProject]:
    LogicalProject(EMPNO=[$0], EXPR$1=[CASE(IS NOT NULL($3), 1:BIGINT, 0:BIGINT)])
      LogicalTableScan(table=[[CATALOG, SALES, EMP]])
```

Hive Case Studies

Hanging TPC-DS queries

- Context: Upgrade Calcite version from 1.25.0 to 1.33.0
- Symptom: TPCDS queries hanging

Hanging TPC-DS queries

- Context: Upgrade Calcite version from 1.25.0 to 1.33.0
- Symptom: TPCDS queries hanging (e.g., query13)

```sql
select avg(ss_quantity),avg(ss_ext_sales_price),avg(ss_ext_wholesale_cost),sum(ss_ext_wholesale_cost)
from store_sales
    ,store
    ,customer_demographics
    ,household_demographics
    ,customer_address
    ,date_dim
where s_store_sk = ss_store_sk and  ss_sold_date_sk = d_date_sk and d_year = 2001 and (
(ss_hdemo_sk=hd_demo_sk and cd_demo_sk = ss_cdemo_sk and cd_marital_status = 'M'
 and cd_education_status = '4 yr Degree' and ss_sales_price between 100.00 and 150.00 and hd_dep_count = 3) or
(ss_hdemo_sk=hd_demo_sk and cd_demo_sk = ss_cdemo_sk and cd_marital_status = 'D'
 and cd_education_status = 'Primary' and ss_sales_price between 50.00 and 100.00 and hd_dep_count = 1) or
(ss_hdemo_sk=hd_demo_sk and cd_demo_sk = ss_cdemo_sk and cd_marital_status = 'U'
 and cd_education_status = 'Advanced Degree' and ss_sales_price between 150.00 and 200.00 and hd_dep_count = 1))
and(
(ss_addr_sk = ca_address_sk and ca_country = 'United States' and ca_state in ('KY', 'GA', 'NM')
 and ss_net_profit between 100 and 200) or
(ss_addr_sk = ca_address_sk and ca_country = 'United States' and ca_state in ('MT', 'OR', 'IN')
 and ss_net_profit between 150 and 300) or
(ss_addr_sk = ca_address_sk and ca_country = 'United States' and ca_state in ('WI', 'MO', 'WV')
 and ss_net_profit between 50 and 250));
```

Hanging TPC-DS queries

- Context: Upgrade Calcite version from 1.25.0 to 1.33.0
- Symptom: TPCDS queries hanging (e.g., query13)

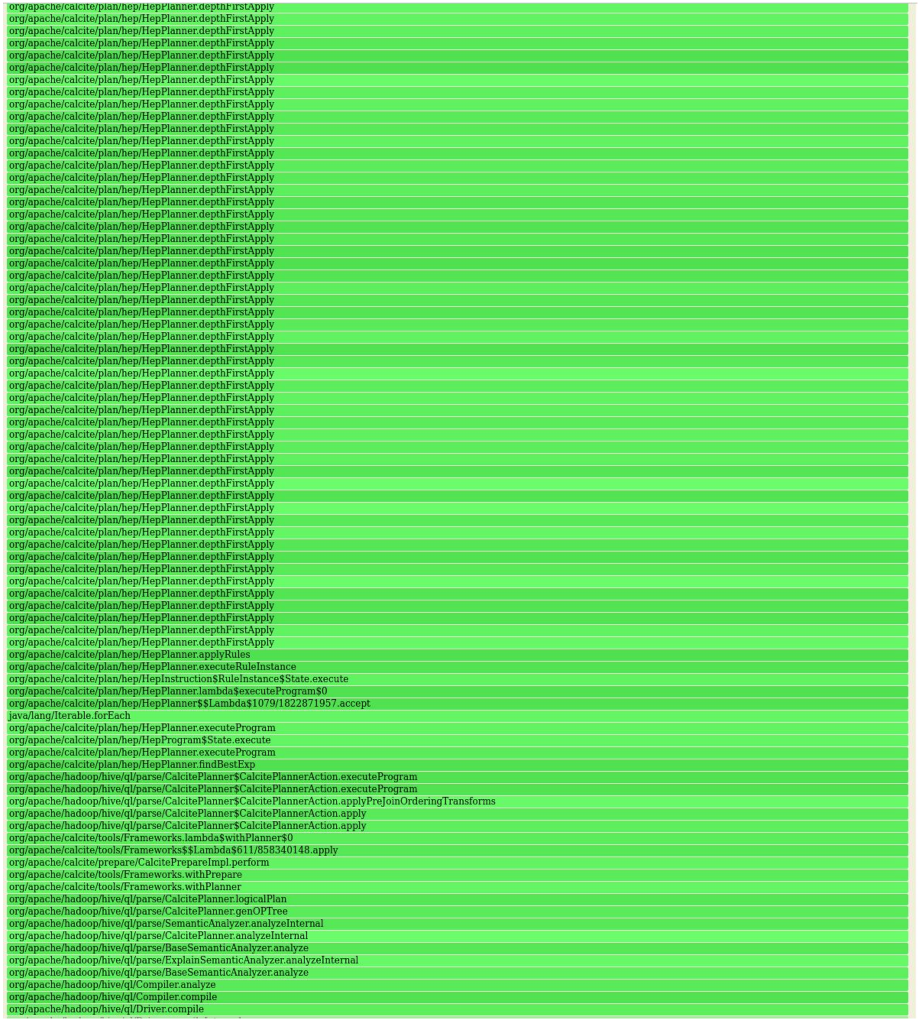
```sql
select avg(ss_quantity),avg(ss_ext_sales_price),avg(ss_ext_wholesale_cost),sum(ss_ext_wholesale_cost)
from store_sales
    ,store
    ,customer_demographics
    ,household_demographics
    ,customer_address
    ,date_dim
where s_store_sk = ss_store_sk and  ss_sold_date_sk = d_date_sk and d_year = 2001 and (
(ss_hdemo_sk=hd_demo_sk and cd_demo_sk = ss_cdemo_sk and cd_marital_status = 'M'
 and cd_education_status = '4 yr Degree' and ss_sales_price between 100.00 and 150.00 and hd_dep_count = 3) or
(ss_hdemo_sk=hd_demo_sk and cd_demo_sk = ss_cdemo_sk and cd_marital_status = 'D'
 and cd_education_status = 'Primary' and ss_sales_price between 50.00 and 100.00 and hd_dep_count = 1) or
(ss_hdemo_sk=hd_demo_sk and cd_demo_sk = ss_cdemo_sk and cd_marital_status = 'U'
 and cd_education_status = 'Advanced Degree' and ss_sales_price between 150.00 and 200.00 and hd_dep_count = 1))
and(
(ss_addr_sk = ca_address_sk and ca_country = 'United States' and ca_state in ('KY', 'GA', 'NM')
 and ss_net_profit between 100 and 200) or
(ss_addr_sk = ca_address_sk and ca_country = 'United States' and ca_state in ('MT', 'OR', 'IN')
 and ss_net_profit between 150 and 300) or
(ss_addr_sk = ca_address_sk and ca_country = 'United States' and ca_state in ('WI', 'MO', 'WV')
 and ss_net_profit between 50 and 250));
```

Hanging TPC-DS queries

- Why does it take so long?
- Where is it stuck?
- What does it do?
- Gather information:
  - Profile the application (async-profiler)
  - Collect stack traces (jstack)
  - Check the logs

Hanging TPC-DS queries

- Why does it take so long?
- Where is it stuck?
- What does it do?
- Gather information:
  - Profile the application (async-profiler)
  - Collect stack traces (jstack)
  - Check the logs



```
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.depthFirstApply
org/apache/calcite/plan/hep/HepPlanner.applyRules
org/apache/calcite/plan/hep/HepPlanner.executeRuleInstance
org/apache/calcite/plan/hep/HepInstruction$RuleInstance$State.execute
org/apache/calcite/plan/hep/HepPlanner.lambda$executeProgram$0
org/apache/calcite/plan/hep/HepPlanner$$Lambda$1079/1822871957.accept
java/lang/Iterable.forEach
org/apache/calcite/plan/hep/HepPlanner.executeProgram
org/apache/calcite/plan/hep/HepProgram$State.execute
org/apache/calcite/plan/hep/HepPlanner.executeProgram
org/apache/calcite/plan/hep/HepPlanner.findBestExp
org/apache/hadoop/hive/ql/parse/CalcitePlanner$CalcitePlannerAction.executeProgram
org/apache/hadoop/hive/ql/parse/CalcitePlanner$CalcitePlannerAction.executeProgram
org/apache/hadoop/hive/ql/parse/CalcitePlanner$CalcitePlannerAction.applyPreJoinOrderingTransforms
org/apache/hadoop/hive/ql/parse/CalcitePlanner$CalcitePlannerAction.apply
org/apache/hadoop/hive/ql/parse/CalcitePlanner$CalcitePlannerAction.apply
org/apache/calcite/tools/Frameworks.lambda$withPlanner$0
org/apache/calcite/tools/Frameworks$$Lambda$611/858340148.apply
org/apache/calcite/prepare/CalcitePrepareImpl.perform
org/apache/calcite/tools/Frameworks.withPrepare
org/apache/calcite/tools/Frameworks.withPlanner
org/apache/hadoop/hive/ql/parse/CalcitePlanner.logicalPlan
org/apache/hadoop/hive/ql/parse/CalcitePlanner.genOPTree
org/apache/hadoop/hive/ql/parse/SemanticAnalyzer.analyzeInternal
org/apache/hadoop/hive/ql/parse/CalcitePlanner.analyzeInternal
org/apache/hadoop/hive/ql/parse/BaseSemanticAnalyzer.analyze
org/apache/hadoop/hive/ql/parse/ExplainSemanticAnalyzer.analyzeInternal
org/apache/hadoop/hive/ql/parse/BaseSemanticAnalyzer.analyze
org/apache/hadoop/hive/ql/Compiler.analyze
org/apache/hadoop/hive/ql/Compiler.compile
org/apache/hadoop/hive/ql/Driver.compile
```

```
java.lang.Thread.State: RUNNABLE
   at java.util.TreeMap.compare(TreeMap.java:1294)
   at java.util.TreeMap.put(TreeMap.java:538)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.replaceRangeWithSameLowerBound(TreeRangeSet.java:272)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.add(TreeRangeSet.java:222)
   at org.apache.hive.com.google.common.collect.RangeSet.addAll(RangeSet.java:225)
   at org.apache.hive.com.google.common.collect.AbstractRangeSet.addAll(AbstractRangeSet.java:64)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.addAll(TreeRangeSet.java:41)
   at org.apache.calcite.rex.RexSimplify$RexSargBuilder.addSarg(RexSimplify.java:3056)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2b(RexSimplify.java:2894)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2(RexSimplify.java:2812)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept_(RexSimplify.java:2793)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept(RexSimplify.java:2778)
   at org.apache.calcite.rex.RexSimplify$SargCollector.access$400(RexSimplify.java:2761)
   at org.apache.calcite.rex.RexSimplify.lambda$simplifyAnd$3(RexSimplify.java:1488)
   at org.apache.calcite.rex.RexSimplify$$Lambda$1099/1247334493.accept(Unknown Source)
   at java.util.ArrayList.forEach(ArrayList.java:1259)
   at org.apache.calcite.rex.RexSimplify.simplifyAnd(RexSimplify.java:1488)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:279)
   at org.apache.calcite.rex.RexSimplify.simplifyUnknownAs(RexSimplify.java:248)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:223)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:299)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
```

```
java.lang.Thread.State: RUNNABLE
   at java.util.TreeMap.compare(TreeMap.java:1294)
   at java.util.TreeMap.put(TreeMap.java:538)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.replaceRangeWithSameLowerBound(TreeRangeSet.java:272)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.add(TreeRangeSet.java:222)
   at org.apache.hive.com.google.common.collect.RangeSet.addAll(RangeSet.java:225)
   at org.apache.hive.com.google.common.collect.AbstractRangeSet.addAll(AbstractRangeSet.java:64)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.addAll(TreeRangeSet.java:41)
   at org.apache.calcite.rex.RexSimplify$RexSargBuilder.addSarg(RexSimplify.java:3056)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2b(RexSimplify.java:2894)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2(RexSimplify.java:2812)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept_(RexSimplify.java:2793)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept(RexSimplify.java:2778)
   at org.apache.calcite.rex.RexSimplify$SargCollector.access$400(RexSimplify.java:2761)
   at org.apache.calcite.rex.RexSimplify.lambda$simplifyAnd$3(RexSimplify.java:1488)
   at org.apache.calcite.rex.RexSimplify$$Lambda$1099/1247334493.accept(Unknown Source)
   at java.util.ArrayList.forEach(ArrayList.java:1259)
   at org.apache.calcite.rex.RexSimplify.simplifyAnd(RexSimplify.java:1488)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:279)
   at org.apache.calcite.rex.RexSimplify.simplifyUnknownAs(RexSimplify.java:248)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:223)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:299)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
```

Depth:~ 2K Lines

```
java.lang.Thread.State: RUNNABLE
   at java.util.TreeMap.compare(TreeMap.java:1294)
   at java.util.TreeMap.put(TreeMap.java:538)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.replaceRangeWithSameLowerBound(TreeRangeSet.java:272)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.add(TreeRangeSet.java:222)
   at org.apache.hive.com.google.common.collect.RangeSet.addAll(RangeSet.java:225)
   at org.apache.hive.com.google.common.collect.AbstractRangeSet.addAll(AbstractRangeSet.java:64)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.addAll(TreeRangeSet.java:41)
   at org.apache.calcite.rex.RexSimplify$RexSargBuilder.addSarg(RexSimplify.java:3056)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2b(RexSimplify.java:2894)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2(RexSimplify.java:2812)
   at org.a
   at org.a
   at org.a
   at org.apache.calcite.rex.RexSimplify.lambda$simplifyAnd$3(RexSimplify.java:1488)
   at org.apache.calcite.rex.RexSimplify$$Lambda$1099/1247334493.accept(Unknown Source)
   at java.util.ArrayList.forEach(ArrayList.java:1259)
   at org.apache.calcite.rex.RexSimplify.simplifyAnd(RexSimplify.java:1488)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:279)
   at org.apache.calcite.rex.RexSimplify.simplifyUnknownAs(RexSimplify.java:248)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:223)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:299)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
```

From stack + code we can infer that the plan has more than 1K nested Filter operators

Depth:~ 2K Lines

```
java.lang.Thread.State: RUNNABLE
   at java.util.TreeMap.compare(TreeMap.java:1294)
   at java.util.TreeMap.put(TreeMap.java:538)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.replaceRangeWithSameLowerBound(TreeRangeSet.java:272)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.add(TreeRangeSet.java:222)
   at org.apache.hive.com.google.common.collect.RangeSet.addAll(RangeSet.java:225)
   at org.apache.hive.com.google.common.collect.AbstractRangeSet.addAll(AbstractRangeSet.java:64)
   at org.apache.hive.com.google.common.collect.TreeRangeSet.addAll(TreeRangeSet.java:41)
   at org.apache.calcite.rex.RexSimplify$RexSargBuilder.addSarg(RexSimplify.java:3056)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2b(RexSimplify.java:2894)
   at org.apache.calcite.rex.RexSimplify$SargCollector.accept2(RexSimplify.java:2812)
   at org.a
   at org.a
   at org.a
   at org.a
   at org.a
   at java.
   at org.apache.calcite.rex.RexSimplify.simplifyAnd(RexSimplify.java:1488)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:279)
   at org.apache.calcite.rex.RexSimplify.simplifyUnknownAs(RexSimplify.java:248)
   at org.apache.calcite.rex.RexSimplify.simplify(RexSimplify.java:223)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:299)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
   at org.apache.calcite.rel.metadata.RelMdPredicates.getPredicates(RelMdPredicates.java:292)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates_$(Unknown Source)
   at org.apache.calcite.rel.metadata.janino.GeneratedMetadata_PredicatesHandler.getPredicates(Unknown Source)
   at org.apache.calcite.rel.metadata.RelMetadataQuery.getPulledUpPredicates(RelMetadataQuery.java:841)
```

From stack + code we can infer that the plan has more than 1K nested Filter operators

What creates the operators? **Why?**

Depth:~ 2K Lines

Hanging TPC-DS queries - Logs to the rescue

```
grep -A 10 "Rule.*produced" hive.log
```

Hanging TPC-DS queries - Logs to the rescue

```
2022-10-07T05:52:23,575 DEBUG calcite.RuleEventLogger: call#1: Rule [HivePreFilteringRule] produced [rel#84:HiveFilter]
2022-10-07T05:52:23,576 DEBUG calcite.RuleEventLogger: call#1: Full plan for [rel#84:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
    HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
      HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveTableScan(table=[[default, store_sales]], table:alias=[store_sales])
              HiveTableScan(table=[[default, store]], table:alias=[store])
--
2022-10-07T05:52:23,601 DEBUG calcite.RuleEventLogger: call#2: Rule [HivePreFilteringRule] produced [rel#89:HiveFilter]
2022-10-07T05:52:23,601 DEBUG calcite.RuleEventLogger: call#2: Full plan for [rel#89:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
    HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
      HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
                HiveTableScan(table=[[default, store_sales]], table:alias=[store_sales])
--
2022-10-07T05:52:23,610 DEBUG calcite.RuleEventLogger: call#3: Rule [HivePreFilteringRule] produced [rel#94:HiveFilter]
2022-10-07T05:52:23,610 DEBUG calcite.RuleEventLogger: call#3: Full plan for [rel#94:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
    HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
      HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
```

Hanging TPC-DS queries - Logs to the rescue

```
2022-10-07T05:52:23,575 DEBUG calcite.RuleEventLogger: call#1: Rule [HivePreFilteringRule] produced [rel#84:HiveFilter]
2022-10-07T05:52:23,576 DEBUG calcite.RuleEventLogger: call#1: Full plan for [rel#84:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
    HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
      HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveTableScan(table=[[default, store_sales]], table:alias=[store_sales])
              HiveTableScan(table=[[default, store]], table:alias=[store])
--
2022-10-07T05:52:23,601 DEBUG calcite.RuleEventLogger: call#2: Rule [HivePreFilteringRule] produced [rel#89:HiveFilter]
2022-10-07T05:52:23,601 DEBUG calcite.RuleEventLogger: call#2: Full plan for [rel#89:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
    HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
      HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
                HiveTableScan(table=[[default, store_sales]], table:alias=[store_sales])
--
2022-10-07T05:52:23,610 DEBUG calcite.RuleEventLogger: call#3: Rule [HivePreFilteringRule] produced [rel#94:HiveFilter]
2022-10-07T05:52:23,610 DEBUG calcite.RuleEventLogger: call#3: Full plan for [rel#94:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
    HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
      HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree         '),
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
```

Hanging TPC-DS queries - Root cause

```
2022-10-07T05:52:23,575 DEBUG calcite.RuleEventLogger: call#1: Rule [HivePreFilteringRule] produced [rel#84:HiveFilter]
2022-10-07T05:52:23,576 DEBUG calcite.RuleEventLogger: call#1: Full plan for [rel#84:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree          '),
    HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
      HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveTableScan(table=[[default, store_sales]], table:alias=[store_sales])
              HiveTableScan(table=[[default, store]], table:alias=[store])
--
2022-10-07T05
2022-10-07T05
HiveFilter(con                                                                                                              _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree          '),
    HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree          '),
      HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
                HiveTableScan(table=[[default, store_sales]], table:alias=[store_sales])
--
2022-10-07T05:52:23,610 DEBUG calcite.RuleEventLogger: call#3: Rule [HivePreFilteringRule] produced [rel#94:HiveFilter]
2022-10-07T05:52:23,610 DEBUG calcite.RuleEventLogger: call#3: Full plan for [rel#94:HiveFilter]:
HiveFilter(condition=[AND(=($27, $6), =($22, $99), =($105, 2001), =($4, $73), =($60, $3), OR(AND(=($62, _UTF-16LE'M'), =($63, _UTF-16LE'4 yr Degree
  HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree          '),
    HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree          '),
      HiveFilter(condition=[AND(OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U')), OR(=($63, _UTF-16LE'4 yr Degree          '),
        HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
          HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
            HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
              HiveJoin(condition=[true], joinType=[inner], algorithm=[none], cost=[not available])
```

HivePreFilteringRule matches infinitely and creates identical filters multiple times

Hanging TPC-DS queries - Interactive debugging

```
OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U'))
```

Hanging TPC-DS queries - Interactive debugging

```
OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U'))
```

```
// 3. If the new conjuncts are already present in the plan, we bail out
final List<RexNode> newConjuncts =
HiveCalciteUtil.getPredsNotPushedAlready(filter.getInput(),operandsToPushDown);
```

Hanging TPC-DS queries - Interactive debugging

```
OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U'))
```

```
// 3. If the new conjuncts are already present in the plan, we bail out
final List<RexNode> newConjuncts =
HiveCalciteUtil.getPredsNotPushedAlready(filter.getInput(),operandsToPushDown);
```

```
mq.getPulledUpPredicates(inp).pulledUpPredicates
```

```
SEARCH($62, Sarg[_UTF-16LE'D', _UTF-16LE'M', _UTF-16LE'U']:CHAR(1))
```

Hanging TPC-DS queries - Interactive debugging

```
OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U'))
```

≠

```
SEARCH($62, Sarg[_UTF-16LE'D', _UTF-16LE'M', _UTF-16LE'U']:CHAR(1))
```

Hanging TPC-DS queries - Interactive debugging

```
OR(=($62, _UTF-16LE'M'), =($62, _UTF-16LE'D'), =($62, _UTF-16LE'U'))
```

≠

```
SEARCH($62, Sarg[_UTF-16LE'D', _UTF-16LE'M', _UTF-16LE'U']:CHAR(1))
```

CALCITE-4173 Add internal SEARCH operator and Sarg literal, replacing use of IN in RexCall (1.26.0)
CALCITE-5036 RelMetadataQuery#getPulledUpPredicates support to analyze constant key for the operator of IS_NOT_DISTINCT_FROM (1.31.0)

Check What CBO Rules Are Applied

Command: `grep --no-filename "produced" hive.log | cut -c 100-`

Output format: `call#$X: Rule [$RULE_DESCRIPTION] produced [rel#$Y:$REL_KIND]`

Sample output:

```
call#1: Rule [HivePreFilteringRule] produced [rel#73:HiveFilter]
call#12: Rule [ReduceExpressionsRule(Filter)] produced [rel#76:HiveFilter]
call#14: Rule [FilterCondition] produced [rel#78:HiveFilter]
call#17: Rule [ReduceExpressionsRule(Filter)] produced [rel#80:HiveFilter]
call#21: Rule [FilterCondition] produced [rel#82:HiveFilter]
call#29: Rule [HiveProjectFilterPullUpConstantsRule] produced [rel#84:HiveProject]
call#46: Rule [HiveJoinAddNotNullRule] produced [rel#89:HiveJoin]
```

Once you have the *rel* number ($Y), you can look it up in the logs to see it printed in full

If you are interested in a particular rule application, search for the *call* ($X), input and output rels are printed close to the corresponding “produced” line

OutOfMemory at Planning Time

- We look for a sequence of rule applications which loops from a certain point of the query planning process
- Run “EXPLAIN” against the offending query and look at the rules which are invoked in the logs to identify the loop
- Once identified, find the first occurrence of applications of this list, and start analyzing the plans generated right before/after it
- Note: the *rel* triggering the loop might also be created outside this sequence

OutOfMemory at Planning Time - Example (HIVE-25758)

```
SELECT c.month, d.con_usd
FROM
  (SELECT
cast(regexp_replace(substr(add_months(from_unixtime(unix_timestamp(),
'yyyy-MM-dd'), -1), 1, 7), '-', '') AS int) AS month
   FROM test1
     UNION ALL
   SELECT month
   FROM test2
   WHERE month = 202110
  ) c
  JOIN test3 d ON c.month = d.mth;
```

- Symptoms: OOM due to the creation of ever increasingly complex filters

OutOfMemory at Planning Time - Example (HIVE-25758)

```
HiveJoin(condition=[=($0, $1)], joinType=[inner], algorithm=[none], cost=[not
available])
  HiveUnion(all=[true])
    HiveProject(month=[CAST(regexp_replace(...):INTEGER**])
      HiveFilter(condition=[IS NOT NULL(CAST(regexp_replace(...):INTEGER)])
        HiveTableScan(table=[[default, test1]], table:alias=[test1])
    HiveProject(month=[CAST(202110):INTEGER])
      HiveFilter(condition=[=($0, 202110)])
        HiveTableScan(table=[[default, test2]], table:alias=[test2])
  HiveFilter(condition=[IS NOT NULL($0)])
    HiveTableScan(table=[[default, test3]], table:alias=[d])
```

```
** Abbreviation of “CAST(regexp_replace(substr(add_months(FROM_UNIXTIME(UNIX_TIMESTAMP,
_UTF-16LE'yyyy-MM-dd':VARCHAR(2147483647)
CHARACTER SET "UTF-16LE"), -1), 1, 7), _UTF-16LE'-':VARCHAR(2147483647) CHARACTER SET
"UTF-16LE", _UTF-16LE'':VARCHAR(2147483647) CHARACTER SET
 "UTF-16LE")):INTEGER”
```

OutOfMemory at Planning Time - Example (HIVE-25758)

RHS → LHS

```
call#198: Rule [HiveJoinPushTransitivePredicatesRule] produced [rel#136:HiveJoin]
call#212: Rule [HiveFilterSetOpTransposeRule] produced [rel#142:HiveUnion]
call#223: Rule [HiveFilterProjectTransposeRule] produced [rel#148:HiveProject]
call#228: Rule [HiveFilterMergeRule] produced [rel#152:HiveFilter]
call#239: Rule [HiveFilterProjectTransposeRule] produced [rel#153:HiveProject]
```

LHS → RHS

```
call#258: Rule [HiveJoinPushTransitivePredicatesRule] produced [rel#155:HiveJoin]
call#273: Rule [HiveFilterProjectTransposeRule] produced [rel#160:HiveProject]
call#278: Rule [HiveFilterMergeRule] produced [rel#164:HiveFilter]
call#283: Rule [ReduceExpressionsRule(Filter)] produced [rel#166:HiveFilter]
```

OutOfMemory at Planning Time - Example (HIVE-25758)

```
HiveJoin(condition=[=($0, $1)], joinType=[inner], [...])
  HiveUnion(all=[true])
    HiveProject(month=[CAST(regexp_replace(...)):INTEGER**])
      HiveFilter(condition=[IS NOT NULL(CAST(regexp_replace(...):INTEGER)])
        HiveTableScan(table=[[default, test1]], table:alias=[test1])
    HiveProject(month=[CAST(202110):INTEGER])
      HiveFilter(condition=[=($0, 202110)])
        HiveTableScan(table=[[default, test2]], table:alias=[test2])
  HiveFilter(condition=[IS NOT NULL($0)])
    HiveTableScan(table=[[default, test3]], table:alias=[d])
```

```
** Abbreviation of “CAST(regexp_replace(substr(add_months(FROM_UNIXTIME(UNIX_TIMESTAMP,
_UTF-16LE'yyyy-MM-dd':VARCHAR(2147483647)
CHARACTER SET "UTF-16LE"), -1), 1, 7), _UTF-16LE'-':VARCHAR(2147483647) CHARACTER SET
"UTF-16LE", _UTF-16LE'':VARCHAR(2147483647) CHARACTER SET
 "UTF-16LE")):INTEGER”
```

OutOfMemory at Planning Time - Example (HIVE-25758)

```
HiveJoin(condition=[=($0, $1)], joinType=[inner], [...])
  HiveUnion(all=[true])
    HiveProject(month=[CAST(regexp_replace(...)):INTEGER])
      HiveFilter(condition=[IS NOT NULL(CAST(regexp_replace(...)):INTEGER)])
        HiveTableScan(table=[[default, test1]], table:alias=[test1])
    HiveProject(month=[CAST(202110):INTEGER])
      HiveFilter(condition=[=($0, 202110)])
        HiveTableScan(table=[[default, test2]], table:alias=[test2])
  HiveFilter(condition=[OR(=($0, CAST(regexp_replace(...)):INTEGER), =($0, 
202110))])
    HiveFilter(condition=[IS NOT NULL($0)])
      HiveTableScan(table=[[default, test3]], table:alias=[d])
```

call#258: Rule [HiveJoinPushTransitivePredicatesRule] produced [rel#155:HiveJoin]

OutOfMemory at Planning Time - Example (HIVE-25758)

```
HiveJoin(condition=[=($0, $1)], joinType=[inner], [...])
  HiveUnion(all=[true])
    HiveProject(month=[CAST(regexp_replace(...)):INTEGER])
      HiveFilter(condition=[IS NOT NULL(CAST(regexp_replace(...)):INTEGER)])
        HiveTableScan(table=[[default, test1]], table:alias=[test1])
    HiveProject(month=[CAST(202110):INTEGER])
      HiveFilter(condition=[=($0, 202110)])
        HiveTableScan(table=[[default, test2]], table:alias=[test2])
  HiveFilter(condition=[AND(OR(=($0, CAST(regexp_replace(...)):INTEGER), =($0,
202110)), IS NOT NULL($0))])
  HiveTableScan(table=[[default, test3]], table:alias=[d])
```

call#278: Rule [HiveFilterMergeRule] produced [rel#164:HiveFilter]

OutOfMemory at Planning Time - Example (HIVE-25758)

```
HiveJoin(condition=[=($0, $1)], joinType=[inner], [...])
  HiveUnion(all=[true])
    HiveProject(month=[CAST(regexp_replace(...)):INTEGER])
      HiveFilter(condition=[IS NOT NULL(CAST(regexp_replace(...)):INTEGER)])
        HiveTableScan(table=[[default, test1]], table:alias=[test1])
    HiveProject(month=[CAST(202110):INTEGER])
      HiveFilter(condition=[=($0, 202110)])
        HiveTableScan(table=[[default, test2]], table:alias=[test2])
  HiveFilter(condition=[AND(OR(=($0, CAST(regexp_replace(...)):INTEGER), =($0,
202110)), IS NOT NULL($0))])
  HiveTableScan(table=[[default, test3]], table:alias=[d])
```

ReduceExpressionsRule(Filter) can’t simplify the predicate

OutOfMemory at Planning Time - Example (HIVE-25758)

```
HiveFilter(condition=[
  AND(
    IN($0, CAST(regexp_replace(...)):INTEGER, 202110),
    OR(
      AND(
        OR(
          IS NOT NULL(CAST(regexp_replace(...)):INTEGER),
          =(CAST(regexp_replace(...)):INTEGER, 202110)
        ),
        =($0, CAST(regexp_replace(...)):INTEGER)
      ),
      =($0, 202110)
    )
  )
])
```

Incomplete / Incorrect Plan

- Usually boils down to identifying the first *rel* that looks “bad”, and trace it back by looking for the *rel*#, which is unique, and should appear in the preceding part of the logs

- Resolution highly depends on the specific issue, but with a precise *call*# or *rel*# it’s possible to set a conditional breakpoint (rules can be invoked multiple times before producing the issue)

Incomplete / Incorrect Plan Hive - Example (HIVE-26722)

- KO: (missing results)

```
SELECT * FROM (
  SELECT a, b FROM t
   UNION ALL
  SELECT a, cast(NULL as STRING) FROM t
) AS t2 WHERE a = 1;
```

```
HiveProject(a=[$0], b=[$1])
  HiveFilter(condition=[=(CAST($0):DOUBLE, 1)])
    HiveTableScan(table=[[default, t]], table:alias=[t])
```

- OK:

```
SELECT * FROM (
  SELECT a, b FROM t
   UNION ALL
  SELECT a, NULL FROM t
) AS t2 WHERE a = 1;
```

```
HiveUnion(all=[true])
  HiveProject(a=[$0], b=[$1])
    HiveFilter(condition=[=(CAST($0):DOUBLE, 1)])
      HiveTableScan(table=[[default, t]], table:alias=[t])
  HiveProject(a=[$0], _o__c1=[null:VARCHAR(2147483647)
CHARACTER SET "UTF-16LE"])
    HiveFilter(condition=[=(CAST($0):DOUBLE, 1)])
      HiveTableScan(table=[[default, t]], table:alias=[t])
```

Incomplete / Incorrect Plan Hive - Example (HIVE-26722)

```
call#5: Rule [HiveFilterProjectTransposeRule]
produced [rel#41:HiveFilter]
 call#6: Rule [HiveFilterSetOpTransposeRule]
produced [rel#43:HiveFilter]
 call#7: Rule [HiveFilterProjectTransposeRule]
produced [rel#47:HiveProject]
 call#24: Rule [HivePartitionPruneRule(Filter)]
produced [rel#52:HiveFilter]
 call#32: Rule [HiveFieldTrimmerRule] produced
[rel#70:HiveFilter]
 call#41: Rule [HiveFilterProjectTSTransposeRule]
produced [rel#77:HiveProject]
 call#43: Rule [HiveCardinalityPreservingJoinRule]
produced [rel#87:HiveProject]
```

```
 call#5: Rule [HiveFilterProjectTransposeRule]
produced [rel#47:HiveFilter]
 call#6: Rule [HiveFilterSetOpTransposeRule]
produced [rel#51:HiveUnion]
 call#7: Rule [HiveFilterProjectTransposeRule]
produced [rel#57:HiveProject]
 call#13: Rule [HiveFilterProjectTransposeRule]
produced [rel#62:HiveProject]
 call#15: Rule [ReduceExpressionsRule(Project)]
produced [rel#66:HiveProject]
 call#19: Rule [HiveFilterProjectTransposeRule]
produced [rel#69:HiveProject]
 call#49: Rule [HivePartitionPruneRule(Filter)]
produced [rel#73:HiveFilter]
 call#64: Rule [HiveFieldTrimmerRule] produced
...
```

Disabling Rules via Configuration

- AbstractRelOptPlanner#setRuleDescExclusionFilter allows to exclude rules based on a regex over their description, even if registered in the planner
- Benefits:
  - It avoids recompiling to exclude some rules from planning while troubleshooting
  - Can be a quick workaround to customers once the faulty rule(s) is identified
  - Can be activated "per-query"
  - Less invasive than disabling CBO altogether (e.g., "hive.cbo.enable=false")
- HIVE-25880: "Add property to exclude CBO rules by a regex on their description"
- Example:

```
set hive.cbo.rule.exclusion.regex=HiveJoinPushTransitivePredicatesRule|HivePreFilteringRule;
```

Conclusion

- Many planning issues boil down to rule transformations making plans bigger and bigger
- Built-in loggers indispensable for fast troubleshooting
- Easy configuration via XML/Property files (Log4j2)
- Configurable verbosity via FULL_PLAN marker
- Common pain points:
  - Using multiple equivalent operators (OR, SEARCH, IN, etc.)
  - Push/Pull predicate logic in various rules
  - Inconsistent simplifications during planning
- Exploit *AbstractRelOptPlanner#setRuleDescExclusionFilter* for quick workarounds and hypotheses verification

Thank you