Item 88
git_comments:

1. Stop pruning for 2 cases:
2. for the case when any of row groups partially matches the filter, matchAllRowGroupsLocal should be set to false
3. whether all row groups of this group scan fully match the filter
4. For the case when group scan has single row group and it was filtered, no need to create new group scan with the same row group.
5. None of the predicates participated in filter pushdown.
6. For the case when newGroupScan wasn't created, the old one may fully match the filter for the case when row group pruning did not happen.
7. If some of the predicates weren't used in the filter, creates new filter with them on top of current scan. Excludes the case when all predicates weren't used in the filter.
8. If current row group fully matches filter, but row group pruning did not happen, remove the filter.

git_commits:

1. **summary:** DRILL-6865: Filter is not removed from the plan when parquet table fully matches the filter
   message: DRILL-6865: Filter is not removed from the plan when parquet table fully matches the filter closes #1552

github_issues:

github_issues_comments:

github_pulls:

1. **title:** DRILL-6865: Query returns wrong result when filter pruning happens
   body: This PR contains two commits: - The first commit contains changes to preserve predicates from filter condition which weren't used for filter pushdown to avoid the case when the filter is pruned. Instead of pruning whole the filter, only predicates which were used in the row group filtering are removed. Please note, that this problem happened only for the case when row group fully matches to the predicates which are used in the filter pushdown. - The second commit contains changes to remove filter from the plan when parquet table has a single row group and fully matches the filter. For problem descriptions please see [DRILL-6865](https://issues.apache.org/jira/browse/DRILL-6865).

github_pulls_comments:

1. +1, LGTM.

github_pulls_reviews:

1. did not happen, remove the filter
2. fully match
3. Why this change is needed?
4. Won't be else enough? Why check that non converted list is smaller?
5. Please add javadoc explaining cases when we want to omit unsupported expressions and when we don't.
6. At least add assert that will ensure that we did have one row group.
7. Maybe filter creation was done before in a loop for the case when we could not build filter form first row group but were able to build filter for the second (for example, if they came from different files)?
8. Please add comment above.
9. Not sure about this change, initially it was added during `visitBooleanOperator`, now you do this for all.
10. Thanks, fixed
11. Thanks, done
12. `applyFilter()` method from the previous code returns `null` if the filter wasn't created from first row group. I agree with you that schema change may break filter pushdown, but currently, we cannot predict that the filter built from one row group will be suitable for other ones.
13. Thanks, done.
14. Thanks, added.
15. We need to convert initial expression to conjunctive normal form, so it will be splitted into predicates more precisely and they will be divided into predicates which are supported by parquet filter pushdown and predicates which aren't.
16. thanks, changed.
17. For the case when `nonConvertedPredList.size() == predList.size()`, none of the predicates participated in filter pushdown, so `call.transformTo()` shouldn't be called for this case.
18. This change was added because now we try to convert every expression, especially arguments of `AND` operator.
19. Please describe cases when we need this flag.
20. This clear, my concern was second condition `} else if (nonConvertedPredList.size() < predList.size())`. Why we cannot use `else` instead?
21. In this case, `else` will include both `nonConvertedPredList.size() < predList.size()` and `nonConvertedPredList.size() == predList.size()` cases, but as I pointed in the comment above, we shouldn't do anything for the last case.
22. **body:** Added to its Javadoc case when it should be used.
    label: documentation

jira_issues:

1. **summary:** Query returns wrong result when filter pruning happens
   description: In DRILL-5796 was implemented removing the filter from the plan when some (or all) row groups of parquet table fully match the filter. For the case when filter has some predicates which parquet filter predicate does not support, they can be omitted for some cases from the resulting filter predicate. When row groups fully match predicates which left in the filter, the whole filter is removed from the plan and the wrong result is returned. Example of the query for reproducing this bug: {code:sql} create table dfs.tmp.`multi/t1` as select * from cp.`tpch/nation.parquet` where n_nationkey > 5; create table dfs.tmp.`multi/t2` as select * from cp.`tpch/nation.parquet` where n_nationkey < 5; select * from dfs.tmp.`multi` where n_nationkey > 5 and n_nationkey like '%10%'; {code} returns {noformat} +-------+--------------+-----------------+--------------+--+ | dir0 | n_nationkey | n_name | n_regionkey | n_comment | +-------+--------------+-----------------+--------------+--+ | t1 | 6 | FRANCE | 3 | refully final requests. regular, ironi | | t1 | 7 | GERMANY | 3 | l platelets. regular accounts x-ray: unusual, regular acco | | t1 | 8 | INDIA | 2 | ss excuses cajole slyly across the packages. deposits print aroun | | t1 | 9 | INDONESIA | 2 | slyly express asymptotes. regular deposits haggle slyly. carefully ironic hockey players sleep blithely. carefull | | t1 | 10 | IRAN | 4 | efully alongside of the slyly final dependencies. | | t1 | 11 | IRAQ | 4 | nic deposits boost atop the quickly final requests? quickly regula | | t1 | 12 | JAPAN | 2 | ously. final, express gifts cajole a | | t1 | 13 | JORDAN | 4 | ic deposits are blithely about the carefully regular pa | | t1 | 14 | KENYA | 0 | pending excuses haggle furiously deposits. pending, express pinto beans wake fluffily past t | | t1 | 15 | MOROCCO | 0 | rns. blithely bold courts among the closely regular packages use furiously bold platelets? | | t1 | 16 | MOZAMBIQUE | 0 | s. ironic, unusual asymptotes wake blithely r | | t1 | 17 | PERU | 1 | platelets. blithely pending dependencies use fluffily across the even pinto beans. carefully silent accoun | | t1 | 18 | CHINA | 2 | c dependencies. furiously express notornis sleep slyly regular accounts. ideas sleep. depos | | t1 | 19 | ROMANIA | 3 | ular asymptotes are about the furious multipliers. express dependencies nag above the ironically ironic account | | t1 | 20 | SAUDI ARABIA | 4 | ts. silent requests haggle. closely express packages sleep across the

blithely | | t1 | 21 | VIETNAM | 2 | hely enticingly express accounts. even, final | | t1 | 22 | RUSSIA | 3 | requests against the platelets use never according to the quickly regular pint | | t1 | 23 | UNITED KINGDOM | 3 | eans boost carefully special requests. accounts are. carefull | | t1 | 24 | UNITED STATES | 1 | y final packages. slow foxes cajole quickly. quickly silent platelets breach ironic accounts. unusual pinto be | +-------+--------------+-----------------+--------------+--+ {noformat} but single row should be returned: {noformat} +-------+--------------+---------+--------------+---+ | dir0 | n_nationkey | n_name | n_regionkey | n_comment | +-------+--------------+---------+--------------+---+ | t1 | 10 | IRAN | 4 | efully alongside of the slyly final dependencies. | +-------+--------------+---------+--------------+---+ {noformat} Filter is removed from the plan, but it contains a predicate which wasn't applied: {noformat} 00-00 Screen : rowType = RecordType(DYNAMIC_STAR **): rowcount = 19.0, cumulative cost = {77.9 rows, 115.9 cpu, 38.0 io, 0.0 network, 0.0 memory}, id = 400 00-01 Project(**=[$0]) : rowType = RecordType(DYNAMIC_STAR **): rowcount = 19.0, cumulative cost = {76.0 rows, 114.0 cpu, 38.0 io, 0.0 network, 0.0 memory}, id = 399 00-02 Project(T1¦¦**=[$0]) : rowType = RecordType(DYNAMIC_STAR T1¦¦**): rowcount = 19.0, cumulative cost = {57.0 rows, 95.0 cpu, 38.0 io, 0.0 network, 0.0 memory}, id = 398 00-03 Project(T1¦¦**=[$0], n_nationkey=[$1]) : rowType = RecordType(DYNAMIC_STAR T1¦¦**, ANY n_nationkey): rowcount = 19.0, cumulative cost = {38.0 rows, 76.0 cpu, 38.0 io, 0.0 network, 0.0 memory}, id = 397 00-04 Scan(table=[[dfs, tmp, multi]], groupscan=[ParquetGroupScan [entries=[ReadEntryWithPath [path=/tmp/multi/t1/0_0_0.parquet]], selectionRoot=file:/tmp/multi, numFiles=1, numRowGroups=1, usedMetadataFile=false, columns=[`**`, `n_nationkey`]]]) : rowType = RecordType(DYNAMIC_STAR **, ANY n_nationkey): rowcount = 19.0, cumulative cost = {19.0 rows, 38.0 cpu, 38.0 io, 0.0 network, 0.0 memory}, id = 396 {noformat} ---- Additionally, a filter is not removed from the plan when parquet table with single row group is queried: {code:sql} create table dfs.tmp.`singleRowGroupTable` as select * from cp.`tpch/nation.parquet`; explain plan for select * from dfs.tmp.`singleRowGroupTable` where n_nationkey > -1; {code} returns plan {noformat} 00-00 Screen 00-01 Project(**=[$0]) 00-02 Project(T0¦¦**=[$0]) 00-03 SelectionVectorRemover 00-04 Filter(condition=[>($1, -1)]) 00-05 Project(T0¦¦**=[$0], n_nationkey=[$1]) 00-06 Scan(table=[[dfs, tmp, singleRowGroupTable]], groupscan=[ParquetGroupScan [entries=[ReadEntryWithPath [path=file:/tmp/singleRowGroupTable]], selectionRoot=file:/tmp/singleRowGroupTable, numFiles=1, numRowGroups=1, usedMetadataFile=false, columns=[`**`, `n_nationkey`]]]) {noformat} *Also, for the case when a table has multiple files, and filter matches all the table, it is not removed from the plan:* {code:sql} select * from dfs.tmp.`multi` where n_nationkey > -1; {code} has plan {noformat} 00-00 Screen : rowType = RecordType(DYNAMIC_STAR **): rowcount = 12.0, cumulative cost = {109.2 rows, 277.2 cpu, 48.0 io, 0.0 network, 0.0 memory}, id = 196 00-01 Project(**=[$0]) : rowType = RecordType(DYNAMIC_STAR **): rowcount = 12.0, cumulative cost = {108.0 rows, 276.0 cpu, 48.0 io, 0.0 network, 0.0 memory}, id = 195 00-02 Project(T0¦¦**=[$0]) : rowType = RecordType(DYNAMIC_STAR T0¦¦**): rowcount = 12.0, cumulative cost = {96.0 rows, 264.0 cpu, 48.0 io, 0.0 network, 0.0 memory}, id = 194 00-03 SelectionVectorRemover : rowType = RecordType(DYNAMIC_STAR T0¦¦**, ANY n_nationkey): rowcount = 12.0, cumulative cost = {84.0 rows, 252.0 cpu, 48.0 io, 0.0 network, 0.0 memory}, id = 193 00-04 Filter(condition=[>($1, -1)]) : rowType = RecordType(DYNAMIC_STAR T0¦¦**, ANY n_nationkey): rowcount = 12.0, cumulative cost = {72.0 rows, 240.0 cpu, 48.0 io, 0.0 network, 0.0 memory}, id = 192 00-05 Project(T0¦¦**=[$0], n_nationkey=[$1]) : rowType = RecordType(DYNAMIC_STAR T0¦¦**, ANY n_nationkey): rowcount = 24.0, cumulative cost = {48.0 rows, 96.0 cpu, 48.0 io, 0.0 network, 0.0 memory}, id = 191 00-06 Scan(table=[[dfs, tmp, multi]], groupscan=[ParquetGroupScan [entries=[ReadEntryWithPath [path=file:/tmp/multi/t2/0_0_0.parquet], ReadEntryWithPath [path=file:/tmp/multi/t1/0_0_0.parquet]], selectionRoot=file:/tmp/multi, numFiles=2, numRowGroups=2, usedMetadataFile=false, columns=[`**`, `n_nationkey`]]]) : rowType = RecordType(DYNAMIC_STAR **, ANY n_nationkey): rowcount = 24.0, cumulative cost = {24.0 rows, 48.0 cpu, 48.0 io, 0.0 network, 0.0 memory}, id = 190 {noformat}

jira_issues_comments:

1. The issue is also reproduced with the following case: {code:sql} select * from dfs.tmp.`multi` where n_nationkey > 5 and n_nationkey/2 < 5 {code}
2. vvysotskyi opened a new pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552 This PR contains two commits: - The first commit contains changes to preserve predicates from filter condition which weren't used for filter pushdown to avoid the case when the filter is pruned. Instead of pruning whole the filter, only predicates which were used in the row group filtering are removed. Please note, that this problem happened only for the case when row group fully matches to the predicates which are used in the filter pushdown. - The second commit contains changes to remove filter from the plan when parquet table has a single row group and fully matches the filter. For problem descriptions please see [DRILL-6865](https://issues.apache.org/jira/browse/DRILL-6865). -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
3. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235960629 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -310,13 +311,60 @@@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili AbstractParquetGroupScan cloneGroupScan = cloneWithFileSelection(qualifiedFilePath); cloneGroupScan.rowGroupInfos = qualifiedRGs; cloneGroupScan.parquetGroupScanStatistics.collect(cloneGroupScan.rowGroupInfos, cloneGroupScan.parquetTableMetadata); + cloneGroupScan.matchAllRowGroups = matchAllRowGroupsLocal; return cloneGroupScan; } catch (IOException e) { logger.warn("Could not apply filter prune due to Exception : {}", e); return null; } } + + /** + * Returns parquet filter predicate built from specified {@code filterExpr}. + * + * @param filterExpr filter expression to build + * @param udfUtilities udf utilities + * @param functionImplementationRegistry context to find drill function holder + * @param optionManager option manager + * @param omitUnsupportedExprs whether expressions which cannot be converted + * may be omitted from the resulting expression + * @return parquet filter predicate + */ + public ParquetFilterPredicate getParquetFilterPredicate(LogicalExpression filterExpr, + UdfUtilities udfUtilities, FunctionImplementationRegistry functionImplementationRegistry, + OptionManager optionManager, boolean omitUnsupportedExprs) { + // used first row group to receive fields list + RowGroupInfo rowGroup = rowGroupInfos.iterator().next(); Review comment: At least add assert that will ensure that we did have one row group. -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
4. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235960462 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java ########## @@ -63,6 +63,7 @@@ static final Logger logger = LoggerFactory.getLogger(ParquetFilterBuilder.class); private final UdfUtilities udfUtilities; + private final boolean omitUnsupportedExprs; Review comment: Please add javadoc explaining cases when we want to omit unsupported expressions and when we don't. -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
5. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235964950 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -310,13 +311,60 @@@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili AbstractParquetGroupScan cloneGroupScan = cloneWithFileSelection(qualifiedFilePath); cloneGroupScan.rowGroupInfos = qualifiedRGs; cloneGroupScan.parquetGroupScanStatistics.collect(cloneGroupScan.rowGroupInfos, cloneGroupScan.parquetTableMetadata); + cloneGroupScan.matchAllRowGroups = matchAllRowGroupsLocal; return cloneGroupScan; } catch (IOException e) { logger.warn("Could not apply filter prune due to Exception : {}", e); return null; } } + + /** + * Returns parquet filter predicate built from specified {@code filterExpr}. + * + * @param filterExpr filter expression to build + * @param udfUtilities udf utilities + * @param functionImplementationRegistry context to find drill function holder + * @param optionManager option manager + * @param omitUnsupportedExprs whether expressions which cannot be converted + * may be omitted from the resulting expression + * @return parquet filter predicate + */ + public ParquetFilterPredicate getParquetFilterPredicate(LogicalExpression filterExpr, Review comment: Maybe filter creation was done before in a loop for the case when we could not build filter form first row group but were able to build filter for the second (for example, if they came from different files)? -- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
6. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235952813 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -134,12 +132,29 @@@ protected void

doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro // get a conjunctions of the filter condition. For each conjunction, if it refers to ITEM or FLATTEN expression // then we could not pushed down. Otherwise, it's qualified to be pushed down. - final List<RexNode> predList = RelOptUtil.conjunctions(condition); + final List<RexNode> predList = RelOptUtil.conjunctions(RexUtil.toCnf(filter.getCluster().getRexBuilder(), condition)); Review comment: Why this change is needed? ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

7. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235949907 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -172,14 +170,30 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro Stopwatch timer = logger.isDebugEnabled() ? Stopwatch.createStarted() : null; - final GroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, + AbstractParquetGroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, optimizerContext.getFunctionRegistry(), optimizerContext.getPlannerSettings().getOptions()); if (timer != null) { logger.debug("Took {} ms to apply filter on parquet row groups. ", timer.elapsed(TimeUnit.MILLISECONDS)); timer.stop(); } - if (newGroupScan == null) { + if (newGroupScan == null) { + if (groupScan.isMatchAllRowGroups()) { + RelNode child = project == null ? scan : project; + // if current row group fully matches filter, + // but row group pruning wasn't happened, removes filter. Review comment: did not happen, remove the filter ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
8. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235966100 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -262,41 +273,31 @@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili Map<SchemaPath, ColumnStatistics> columnStatisticsMap = statCollector.collectColStat(schemaPathsInExpr); - if (filterPredicate == null) { - ErrorCollector errorCollector = new ErrorCollectorImpl(); - LogicalExpression materializedFilter = ExpressionTreeMaterializer.materializeFilterExpr(- filterExpr, columnStatisticsMap, errorCollector, functionImplementationRegistry); - - if (errorCollector.hasErrors()) { - logger.error("{} error(s) encountered when materialize filter expression : {}", - errorCollector.getErrorCount(), errorCollector.toErrorString()); - return null; - } - logger.debug("materializedFilter : {}", ExpressionStringBuilder.toString(materializedFilter)); - - Set<LogicalExpression> constantBoundaries = ConstantExpressionIdentifier.getConstantExpressionSet(materializedFilter); - filterPredicate = ParquetFilterBuilder.buildParquetFilterPredicate(materializedFilter, constantBoundaries, udfUtilities); - - if (filterPredicate == null) { - return null; - } - } - ParquetFilterPredicate.RowsMatch match = ParquetRGFilterEvaluator.matches(filterPredicate, columnStatisticsMap, rowGroup.getRowCount(), parquetTableMetadata, rowGroup.getColumns(), schemaPathsInExpr); + ParquetFilterPredicate.RowsMatch match = ParquetRGFilterEvaluator.matches(filterPredicate, columnStatisticsMap, rowGroup.getRowCount(), parquetTableMetadata, rowGroup.getColumns(), schemaPathsInExpr); if (match == ParquetFilterPredicate.RowsMatch.NONE) { continue; // No row comply to the filter => drop the row group } rowGroup.setRowsMatch(match); + if (matchAllRowGroupsLocal) { Review comment: Please add comment above. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
9. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235950497 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -85,6 +85,8 @@ private List<EndpointAffinity> endpointAffinities; private ParquetGroupScanStatistics parquetGroupScanStatistics; + // whether all row groups of this group scan fully matches the filter Review comment: fully match ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
10. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235969168 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java ########## @@ -71,18 +72,24 @@ * * @return parquet filter predicate */ - public static ParquetFilterPredicate buildParquetFilterPredicate(LogicalExpression expr, final Set<LogicalExpression> constantBoundaries, UdfUtilities udfUtilities) { - LogicalExpression logicalExpression = expr.accept(new ParquetFilterBuilder(udfUtilities), constantBoundaries); + public static ParquetFilterPredicate buildParquetFilterPredicate(LogicalExpression expr, + Set<LogicalExpression> constantBoundaries, UdfUtilities udfUtilities, + boolean omitUnsupportedExprs) { + LogicalExpression logicalExpression = expr.accept(new ParquetFilterBuilder(udfUtilities, omitUnsupportedExprs), constantBoundaries); if (logicalExpression instanceof ParquetFilterPredicate) { return (ParquetFilterPredicate) logicalExpression; } else if (logicalExpression instanceof TypedFieldExpr) { + // Calcite simplifies `= true` expression to field name, wrap it with is true predicate + return (ParquetFilterPredicate) ParquetIsPredicate.createIsPredicate(FunctionGenerationHelper.IS_TRUE, logicalExpression); Review comment: Not sure about this change, initially it was added during `visitBooleanOperator`, now you do this for all. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
11. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235957943 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -155,39 +170,53 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro Stopwatch timer = logger.isDebugEnabled() ? Stopwatch.createStarted() : null; - final GroupScan newGroupScan = groupScan.applyFilter(conditionExp,optimizerContext, + AbstractParquetGroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, optimizerContext.getFunctionRegistry(), optimizerContext.getPlannerSettings().getOptions()); if (timer != null) { logger.debug("Took {} ms to apply filter on parquet row groups. ", timer.elapsed(TimeUnit.MILLISECONDS)); timer.stop(); } - if (newGroupScan == null) { + if (newGroupScan == null) { + if (groupScan.isMatchAllRowGroups()) { + RelNode child = project == null ? scan : project; + // if current row group fully matches filter, + // but row group pruning wasn't happened, removes filter. + if (nonConvertedPredList.size() == 0) { + call.transformTo(child); + } else if (nonConvertedPredList.size() < predList.size()) { Review comment: Won't be else enough? Why check that non converted list is smaller? ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
12. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235994025 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -85,6 +85,8 @@ private List<EndpointAffinity> endpointAffinities; private ParquetGroupScanStatistics parquetGroupScanStatistics; + // whether all row groups of this group scan fully matches the filter Review comment: Thanks, fixed ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
13. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235994435 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -262,41 +273,31 @@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili Map<SchemaPath, ColumnStatistics> columnStatisticsMap = statCollector.collectColStat(schemaPathsInExpr); - if (filterPredicate == null) { - ErrorCollector errorCollector = new ErrorCollectorImpl(); - LogicalExpression materializedFilter = ExpressionTreeMaterializer.materializeFilterExpr(- filterExpr, columnStatisticsMap, errorCollector, functionImplementationRegistry); - - if (errorCollector.hasErrors()) { - logger.error("{} error(s) encountered when materialize filter expression : {}", - errorCollector.getErrorCount(), errorCollector.toErrorString()); - return null; - } - logger.debug("materializedFilter : {}", ExpressionStringBuilder.toString(materializedFilter)); - - Set<LogicalExpression> constantBoundaries = ConstantExpressionIdentifier.getConstantExpressionSet(materializedFilter); - filterPredicate =

ParquetFilterBuilder.buildParquetFilterPredicate(materializedFilter, constantBoundaries, udfUtilities); - - if (filterPredicate == null) { - return null; - } - } - - ParquetFilterPredicate.RowsMatch match = ParquetRGFilterEvaluator.matches(filterPredicate, columnStatisticsMap, rowGroup.getRowCount(), parquetTableMetadata, rowGroup.getColumns(), schemaPathsInExpr); + ParquetFilterPredicate.RowsMatch match = ParquetRGFilterEvaluator.matches(filterPredicate, + columnStatisticsMap, rowGroup.getRowCount(), parquetTableMetadata, rowGroup.getColumns(), schemaPathsInExpr); if (match == ParquetFilterPredicate.RowsMatch.NONE) { continue; // No row comply to the filter => drop the row group } - rowGroup.setRowsMatch(match); + if (matchAllRowGroupsLocal) { Review comment: Thanks, done ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

14. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235995816 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -310,13 +311,60 @@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili AbstractParquetGroupScan cloneGroupScan = cloneWithFileSelection(qualifiedFilePath); cloneGroupScan.rowGroupInfos = qualifiedRGs; cloneGroupScan.parquetGroupScanStatistics.collect(cloneGroupScan.rowGroupInfos, cloneGroupScan.parquetTableMetadata); + cloneGroupScan.matchAllRowGroups = matchAllRowGroupsLocal; return cloneGroupScan; } catch (IOException e) { logger.warn("Could not apply filter prune due to Exception : {}", e); return null; } } + + /** + * Returns parquet filter predicate built from specified {@code filterExpr}. + * + * @param filterExpr filter expression to build + * @param udfUtilities udf utilities + * @param functionImplementationRegistry context to find drill function holder + * @param optionManager option manager + * @param omitUnsupportedExprs whether expressions which cannot be converted + * may be omitted from the resulting expression + * @return parquet filter predicate + */ + public ParquetFilterPredicate getParquetFilterPredicate(LogicalExpression filterExpr, + UdfUtilities udfUtilities, FunctionImplementationRegistry functionImplementationRegistry, + OptionManager optionManager, boolean omitUnsupportedExprs) { + // used first row group to receive fields list + RowGroupInfo rowGroup = rowGroupInfos.iterator().next(); Review comment: Thanks, done. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

15. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235997768 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java ########## @@ -71,18 +72,24 @@ * * @return parquet filter predicate */ - public static ParquetFilterPredicate buildParquetFilterPredicate(LogicalExpression expr, final Set<LogicalExpression> constantBoundaries, UdfUtilities udfUtilities) { - LogicalExpression logicalExpression = expr.accept(new ParquetFilterBuilder(udfUtilities), constantBoundaries); + public static ParquetFilterPredicate buildParquetFilterPredicate(LogicalExpression expr, + Set<LogicalExpression> constantBoundaries, UdfUtilities udfUtilities, + boolean omitUnsupportedExprs) { + LogicalExpression logicalExpression = + expr.accept(new ParquetFilterBuilder(udfUtilities, omitUnsupportedExprs), constantBoundaries); if (logicalExpression instanceof ParquetFilterPredicate) { return (ParquetFilterPredicate) logicalExpression; } else if (logicalExpression instanceof TypedFieldExpr) { + // Calcite simplifies `= true` expression to field name, wrap it with is true predicate return (ParquetFilterPredicate) ParquetIsPredicate.createIsPredicate(FunctionGenerationHelper.IS_TRUE, logicalExpression); Review comment: This change was added because now we try to convert every expression, especially arguments of `AND` operator. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

16. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235995999 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java ########## @@ -63,6 +63,7 @@ static final Logger logger = LoggerFactory.getLogger(ParquetFilterBuilder.class); private final UdfUtilities udfUtilities; + private final boolean omitUnsupportedExprs; Review comment: Thanks, added. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

17. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235995661 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java ########## @@ -310,13 +311,60 @@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili AbstractParquetGroupScan cloneGroupScan = cloneWithFileSelection(qualifiedFilePath); cloneGroupScan.rowGroupInfos = qualifiedRGs; cloneGroupScan.parquetGroupScanStatistics.collect(cloneGroupScan.rowGroupInfos, cloneGroupScan.parquetTableMetadata); + cloneGroupScan.matchAllRowGroups = matchAllRowGroupsLocal; return cloneGroupScan; } catch (IOException e) { logger.warn("Could not apply filter prune due to Exception : {}", e); return null; } } + + /** + * Returns parquet filter predicate built from specified {@code filterExpr}. + * + * @param filterExpr filter expression to build + * @param udfUtilities udf utilities + * @param functionImplementationRegistry context to find drill function holder + * @param optionManager option manager + * @param omitUnsupportedExprs whether expressions which cannot be converted + * may be omitted from the resulting expression + * @return parquet filter predicate + */ + public ParquetFilterPredicate getParquetFilterPredicate(LogicalExpression filterExpr, Review comment: `applyFilter()` method from the previous code returns `null` if the filter wasn't created from first row group. I agree with you that schema change may break filter pushdown, but currently, we cannot predict that the filter built from one row group will be suitable for other ones. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

18. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235997009 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -134,12 +132,29 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro // get a conjunctions of the filter condition. For each conjunction, if it refers to ITEM or FLATTEN expression // then we could not pushed down. Otherwise, it's qualified to be pushed down. - final List<RexNode> predList = RelOptUtil.conjunctions(condition); + final List<RexNode> predList = RelOptUtil.conjunctions(RexUtil.toCnf(filter.getCluster().getRexBuilder(), condition)); Review comment: We need to convert initial expression to conjunctive normal form, so it will be splitted into predicates more precisely and they will be divided into predicates which are supported by parquet filter pushdown and predicates which aren't. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

19. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235997054 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -172,14 +170,30 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro Stopwatch timer = logger.isDebugEnabled() ? Stopwatch.createStarted() : null; - final GroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, + AbstractParquetGroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, optimizerContext.getFunctionRegistry(), optimizerContext.getPlannerSettings().getOptions()); if (timer != null) { logger.debug("Took {} ms to apply filter on parquet row groups. ", timer.elapsed(TimeUnit.MILLISECONDS)); timer.stop(); } - if (newGroupScan == null) { + if (newGroupScan == null) { + if (groupScan.isMatchAllRowGroups()) { + RelNode child = project == null ? scan : project; + // if current row group fully matches filter, + // but row group pruning wasn't happened, removes filter. Review comment: thanks, changed. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

20. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r235997432 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -155,39 +170,53 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro Stopwatch timer = logger.isDebugEnabled() ? Stopwatch.createStarted() : null; - final GroupScan newGroupScan = groupScan.applyFilter(conditionExp,optimizerContext, + AbstractParquetGroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, optimizerContext.getFunctionRegistry(), optimizerContext.getPlannerSettings().getOptions()); if

(timer != null) { logger.debug("Took {} ms to apply filter on parquet row groups. ", timer.elapsed(TimeUnit.MILLISECONDS)); timer.stop(); } - if (newGroupScan == null) { + if (newGroupScan == null) { + if (groupScan.isMatchAllRowGroups()) { + RelNode child = project == null ? scan : project; + // if current row group fully matches filter, + // but row group pruning wasn't happened, removes filter. + if (nonConvertedPredList.size() == 0) { + call.transformTo(child); + } else if (nonConvertedPredList.size() < predList.size()) { Review comment: For the case when `nonConvertedPredList.size() == predList.size()`, none of the predicates participated in filter pushdown, so `call.transformTo()` shouldn't be called for this case. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

21. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r236044637 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java ########## @@ -63,6 +63,7 @@ static final Logger logger = LoggerFactory.getLogger(ParquetFilterBuilder.class); private final UdfUtilities udfUtilities; + private final boolean omitUnsupportedExprs; Review comment: Please describe cases when we need this flag. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
22. arina-ielchiieva commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r236044696 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -155,39 +170,53 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro Stopwatch timer = logger.isDebugEnabled() ? Stopwatch.createStarted() : null; - final GroupScan newGroupScan = groupScan.applyFilter(conditionExp,optimizerContext, + AbstractParquetGroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, optimizerContext.getFunctionRegistry(), optimizerContext.getPlannerSettings().getOptions()); if (timer != null) { logger.debug("Took {} ms to apply filter on parquet row groups. ", timer.elapsed(TimeUnit.MILLISECONDS)); timer.stop(); } - if (newGroupScan == null) { + if (newGroupScan == null) { + if (groupScan.isMatchAllRowGroups()) { + RelNode child = project == null ? scan : project; + // if current row group fully matches filter, + // but row group pruning wasn't happened, removes filter. + if (nonConvertedPredList.size() == 0) { + call.transformTo(child); + } else if (nonConvertedPredList.size() < predList.size()) { Review comment: This clear, my concern was second condition `} else if (nonConvertedPredList.size() < predList.size())`. Why we cannot use `else` instead? ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
23. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r236069655 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java ########## @@ -155,39 +170,53 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter, ProjectPrel pro Stopwatch timer = logger.isDebugEnabled() ? Stopwatch.createStarted() : null; - final GroupScan newGroupScan = groupScan.applyFilter(conditionExp,optimizerContext, + AbstractParquetGroupScan newGroupScan = groupScan.applyFilter(conditionExp, optimizerContext, optimizerContext.getFunctionRegistry(), optimizerContext.getPlannerSettings().getOptions()); if (timer != null) { logger.debug("Took {} ms to apply filter on parquet row groups. ", timer.elapsed(TimeUnit.MILLISECONDS)); timer.stop(); } - if (newGroupScan == null) { + if (newGroupScan == null) { + if (groupScan.isMatchAllRowGroups()) { + RelNode child = project == null ? scan : project; + // if current row group fully matches filter, + // but row group pruning wasn't happened, removes filter. + if (nonConvertedPredList.size() == 0) { + call.transformTo(child); + } else if (nonConvertedPredList.size() < predList.size()) { Review comment: In this case, `else` will include both `nonConvertedPredList.size() < predList.size()` and `nonConvertedPredList.size() == predList.size()` cases, but as I pointed in the comment above, we shouldn't do anything for the last case. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
24. vvysotskyi commented on a change in pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#discussion_r236069944 ########## File path: exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java ########## @@ -63,6 +63,7 @@ static final Logger logger = LoggerFactory.getLogger(ParquetFilterBuilder.class); private final UdfUtilities udfUtilities; + private final boolean omitUnsupportedExprs; Review comment: Added to its Javadoc case when it should be used. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
25. arina-ielchiieva commented on issue #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552#issuecomment-441604891 +1, LGTM. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
26. asfgit closed pull request #1552: DRILL-6865: Query returns wrong result when filter pruning happens URL: https://github.com/apache/drill/pull/1552 This is a PR merged from a forked repository. As GitHub hides the original diff on merge, it is displayed below for the sake of provenance: As this is a foreign pull request (from a fork), the diff is supplied below (as it won't show otherwise due to GitHub magic): diff --git a/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java b/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java index 9bc969f035b..5699d4546f7 100644 --- a/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java +++ b/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/AbstractParquetGroupScan.java @@ -85,6 +85,8 @@ private List<EndpointAffinity> endpointAffinities; private ParquetGroupScanStatistics parquetGroupScanStatistics; + // whether all row groups of this group scan fully match the filter + private boolean matchAllRowGroups = false; protected AbstractParquetGroupScan(String userName, List<SchemaPath> columns, @@ -111,6 +113,7 @@ protected AbstractParquetGroupScan(AbstractParquetGroupScan that) { this.fileSet = that.fileSet == null ? null : new HashSet<>(that.fileSet); this.entries = that.entries == null ? null : new ArrayList<>(that.entries); this.readerConfig = that.readerConfig; + this.matchAllRowGroups = that.matchAllRowGroups; } @JsonProperty @@ -135,6 +138,11 @@ public ParquetReaderConfig getReaderConfig() { return readerConfig; } + @JsonIgnore + public boolean isMatchAllRowGroups() { + return matchAllRowGroups; + } + @JsonIgnore @Override public Collection<String> getFiles() { @@ -229,15 +237,12 @@ public void setFilter(LogicalExpression filter) { } @Override - public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtilities, - FunctionImplementationRegistry functionImplementationRegistry, OptionManager optionManager) { - if (rowGroupInfos.size() == 1 || - (parquetTableMetadata.isRowGroupPrunable()) || - rowGroupInfos.size() > optionManager.getOption(PlannerSettings.PARQUET_ROWGROUP_FILTER_PUSHDOWN_PLANNING_THRESHOLD) -) { - // Stop pruning for 3 cases: - // - 1 single parquet file, + public AbstractParquetGroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtilities, + FunctionImplementationRegistry functionImplementationRegistry, OptionManager optionManager) { + if (!parquetTableMetadata.isRowGroupPrunable() || + rowGroupInfos.size() > optionManager.getOption(PlannerSettings.PARQUET_ROWGROUP_FILTER_PUSHDOWN_PLANNING_THRESHOLD)) { + // Stop pruning for 2 cases: // - metadata does not have proper format to support row group level filter pruning, // - # of row groups is beyond PARQUET_ROWGROUP_FILTER_PUSHDOWN_PLANNING_THRESHOLD. return null; @@ -248,7 +253,13 @@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili final List<RowGroupInfo> qualifiedRGs = new ArrayList<>(rowGroupInfos.size()); Set<String> qualifiedFilePath = new HashSet<>(); // HashSet keeps a fileName unique. - ParquetFilterPredicate filterPredicate = null; + ParquetFilterPredicate filterPredicate = getParquetFilterPredicate(filterExpr, udfUtilities, functionImplementationRegistry, optionManager, true); + + if (filterPredicate == null) { + return null; + } + + boolean matchAllRowGroupsLocal = true; for (RowGroupInfo rowGroup : rowGroupInfos) { final ColumnExplorer columnExplorer = new ColumnExplorer(optionManager, columns); @@ -262,41 +273,33 @@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili Map<SchemaPath, ColumnStatistics> columnStatisticsMap = statCollector.collectColStat(schemaPathsInExpr); - if (filterPredicate == null) { - ErrorCollector errorCollector = new ErrorCollectorImpl(); - LogicalExpression materializedFilter = ExpressionTreeMaterializer.materializeFilterExpr(- filterExpr, columnStatisticsMap, errorCollector, functionImplementationRegistry); - - if (errorCollector.hasErrors()) { - logger.error("{} error(s) encountered when materialize filter expression : {}", - errorCollector.getErrorCount(), errorCollector.toErrorString()); - return null; - } - logger.debug("materializedFilter : {}", - ExpressionStringBuilder.toString(materializedFilter)); - - Set<LogicalExpression> constantBoundaries =

```
ConstantExpressionIdentifier.getConstantExpressionSet(materializedFilter); - filterPredicate =
ParquetFilterBuilder.buildParquetFilterPredicate(materializedFilter, constantBoundaries, udfUtilities); - - if (filterPredicate == null) { - return null; - } - } - -
ParquetFilterPredicate.RowsMatch match = ParquetRGFilterEvaluator.matches(filterPredicate, columnStatisticsMap, rowGroup.getRowCount(),
parquetTableMetadata, rowGroup.getColumns(), schemaPathsInExpr); + ParquetFilterPredicate.RowsMatch match =
ParquetRGFilterEvaluator.matches(filterPredicate, + columnStatisticsMap, rowGroup.getRowCount(), parquetTableMetadata, rowGroup.getColumns(),
schemaPathsInExpr); if (match == ParquetFilterPredicate.RowsMatch.NONE) { continue; // No row comply to the filter => drop the row group } -
rowGroup.setRowsMatch(match); + // for the case when any of row groups partially matches the filter, + // matchAllRowGroupsLocal should be set to false
+ if (matchAllRowGroupsLocal) { + matchAllRowGroupsLocal = match == ParquetFilterPredicate.RowsMatch.ALL; + } qualifiedRGs.add(rowGroup);
qualifiedFilePath.add(rowGroup.getPath()); } - if (qualifiedRGs.size() == rowGroupInfos.size() ) { + if (qualifiedRGs.size() == rowGroupInfos.size()) { // There is no reduction of rowGroups. Return the original groupScan. logger.debug("applyFilter does not have any pruning!"); + matchAllRowGroups =
matchAllRowGroupsLocal; return null; } else if (qualifiedFilePath.size() == 0) { + if (rowGroupInfos.size() == 1) { + // For the case when group scan has
single row group and it was filtered, + // no need to create new group scan with the same row group. + return null; + } + matchAllRowGroupsLocal = false;
logger.debug("All rowgroups have been filtered out. Add back one to get schema from scannner"); RowGroupInfo rg = rowGroupInfos.iterator().next();
qualifiedFilePath.add(rg.getPath()); @@ -310,6 +313,7 @@ public GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili
AbstractParquetGroupScan cloneGroupScan = cloneWithFileSelection(qualifiedFilePath); cloneGroupScan.rowGroupInfos = qualifiedRGs;
cloneGroupScan.parquetGroupScanStatistics.collect(cloneGroupScan.rowGroupInfos, cloneGroupScan.parquetTableMetadata); +
cloneGroupScan.matchAllRowGroups = matchAllRowGroupsLocal; return cloneGroupScan; } catch (IOException e) { @@ -317,6 +321,53 @@ public
GroupScan applyFilter(LogicalExpression filterExpr, UdfUtilities udfUtili return null; } } + + /** + * Returns parquet filter predicate built from specified
{@code filterExpr}. + * + * @param filterExpr filter expression to build + * @param udfUtilities udf utilities + * @param functionImplementationRegistry
context to find drill function holder + * @param optionManager option manager + * @param omitUnsupportedExprs whether expressions which cannot be
converted + * may be omitted from the resulting expression + * @return parquet filter predicate + */ + public ParquetFilterPredicate
getParquetFilterPredicate(LogicalExpression filterExpr, + UdfUtilities udfUtilities, FunctionImplementationRegistry functionImplementationRegistry, +
OptionManager optionManager, boolean omitUnsupportedExprs) { + // used first row group to receive fields list + assert rowGroupInfos.size() > 0 : "row
groups count cannot be 0"; + RowGroupInfo rowGroup = rowGroupInfos.iterator().next(); + ColumnExplorer columnExplorer = new
ColumnExplorer(optionManager, columns); + + Map<String, String> implicitColValues = columnExplorer.populateImplicitColumns( +
rowGroup.getPath(), + getPartitionValues(rowGroup), + supportsFileImplicitColumns()); + + ParquetMetaStatCollector statCollector = new
ParquetMetaStatCollector( + parquetTableMetadata, + rowGroup.getColumns(), + implicitColValues); + + Set<SchemaPath> schemaPathsInExpr =
filterExpr.accept(new ParquetRGFilterEvaluator.FieldReferenceFinder(), null); + Map<SchemaPath, ColumnStatistics> columnStatisticsMap =
statCollector.collectColStat(schemaPathsInExpr); + + ErrorCollector errorCollector = new ErrorCollectorImpl(); + LogicalExpression materializedFilter =
ExpressionTreeMaterializer.materializeFilterExpr( + filterExpr, columnStatisticsMap, errorCollector, functionImplementationRegistry); + + if
(errorCollector.hasErrors()) { + logger.error("{} error(s) encountered when materialize filter expression : {}", + errorCollector.getErrorCount(),
errorCollector.toErrorString()); + return null; + } + logger.debug("materializedFilter : {}", ExpressionStringBuilder.toString(materializedFilter)); + +
Set<LogicalExpression> constantBoundaries = ConstantExpressionIdentifier.getConstantExpressionSet(materializedFilter); + return
ParquetFilterBuilder.buildParquetFilterPredicate(materializedFilter, constantBoundaries, udfUtilities, omitUnsupportedExprs); + } // filter push down
methods block end // limit push down methods start diff --git a/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java
b/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java index f0f1029119e..86e207fa287 100644 --- a/exec/java-
exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java +++ b/exec/java-
exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetFilterBuilder.java @@ -63,6 +63,12 @@ static final Logger logger =
LoggerFactory.getLogger(ParquetFilterBuilder.class); private final UdfUtilities udfUtilities; + // Flag to check whether predicate cannot be fully converted +
// to parquet filter predicate without omitting its parts. + // It should be set to false for the case when we want to + // verify that predicate is fully convertible
to parquet filter predicate, + // otherwise null is returned instead of the converted expression. + private final boolean omitUnsupportedExprs; /** * @param
expr materialized filter expression @@ -71,18 +77,24 @@ * * @return parquet filter predicate */ - public static ParquetFilterPredicate
buildParquetFilterPredicate(LogicalExpression expr, final Set<LogicalExpression> constantBoundaries, UdfUtilities udfUtilities) { - LogicalExpression
logicalExpression = expr.accept(new ParquetFilterBuilder(udfUtilities), constantBoundaries); + public static ParquetFilterPredicate
buildParquetFilterPredicate(LogicalExpression expr, + Set<LogicalExpression> constantBoundaries, UdfUtilities udfUtilities, boolean
omitUnsupportedExprs) { + LogicalExpression logicalExpression = + expr.accept(new ParquetFilterBuilder(udfUtilities, omitUnsupportedExprs),
constantBoundaries); if (logicalExpression instanceof ParquetFilterPredicate) { return (ParquetFilterPredicate) logicalExpression; + } else if
(logicalExpression instanceof TypedFieldExpr) { + // Calcite simplifies `= true` expression to field name, wrap it with is true predicate + return
(ParquetFilterPredicate) ParquetIsPredicate.createIsPredicate(FunctionGenerationHelper.IS_TRUE, logicalExpression); } logger.debug("Logical expression
{} was not qualified for filter push down", logicalExpression); return null; } - private ParquetFilterBuilder(UdfUtilities udfUtilities) { + private
ParquetFilterBuilder(UdfUtilities udfUtilities, boolean omitUnsupportedExprs) { this.udfUtilities = udfUtilities; + this.omitUnsupportedExprs =
omitUnsupportedExprs; } @Override @@ -159,8 +171,9 @@ public LogicalExpression visitBooleanOperator(BooleanOperator op, Set<LogicalExp for
(LogicalExpression arg : op.args) { LogicalExpression childPredicate = arg.accept(this, value); if (childPredicate == null) { - if
(functionName.equals("booleanOr")) { + if (functionName.equals("booleanOr") || !omitUnsupportedExprs) { // we can't include any leg of the OR if any of
the predicates cannot be converted + // or prohibited omitting of unconverted operands return null; } } else { diff --git a/exec/java-
exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java b/exec/java-
exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java index c59cdce8060..c12ea73129b 100644 --- a/exec/java-
exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java +++ b/exec/java-
exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetPushDownFilter.java @@ -28,9 +28,7 @@ import
org.apache.drill.common.expression.LogicalExpression; import org.apache.drill.common.expression.ValueExpressions; import
org.apache.drill.exec.expr.stat.ParquetFilterPredicate; -import org.apache.drill.exec.expr.stat.ParquetFilterPredicate.RowsMatch; import
org.apache.drill.exec.ops.OptimizerRulesContext; -import org.apache.drill.exec.physical.base.GroupScan; import
org.apache.drill.exec.planner.common.DrillRelOptUtil; import org.apache.drill.exec.planner.logical.DrillOptiq; import
org.apache.drill.exec.planner.logical.DrillParseContext; @@ -134,13 +132,32 @@ protected void doOnMatch(RelOptRuleCall call, FilterPrel filter,
ProjectPrel pro // get a conjunctions of the filter condition. For each conjunction, if it refers to ITEM or FLATTEN expression // then we could not pushed
down. Otherwise, it's qualified to be pushed down. - final List<RexNode> predList = RelOptUtil.conjunctions(condition); + final List<RexNode> predList =
RelOptUtil.conjunctions(RexUtil.toCnf(filter.getCluster().getRexBuilder(), condition)); final List<RexNode> qualifiedPredList = new ArrayList<>(); - for
(final RexNode pred : predList) { + // list of predicates which cannot be converted to parquet filter predicate + List<RexNode> nonConvertedPredList = new
ArrayList<>(); + + for (RexNode pred : predList) { if (DrillRelOptUtil.findOperators(pred, Collections.emptyList(), BANNED_OPERATORS) == null) { +
LogicalExpression drillPredicate = DrillOptiq.toDrill( + new DrillParseContext(PrelUtil.getPlannerSettings(call.getPlanner())), scan, pred); + + // checks
whether predicate may be used for filter pushdown + ParquetFilterPredicate parquetFilterPredicate = + groupScan.getParquetFilterPredicate(drillPredicate, +
optimizerContext, + optimizerContext.getFunctionRegistry(), + optimizerContext.getPlannerSettings().getOptions(), false); + // collects predicates that
contain unsupported for filter pushdown expressions + // to build filter with them + if (parquetFilterPredicate == null) { + nonConvertedPredList.add(pred);
+ } qualifiedPredList.add(pred); + } else { + nonConvertedPredList.add(pred); } } @@ -155,39 +172,58 @@ protected void doOnMatch(RelOptRuleCall
call, FilterPrel filter, ProjectPrel pro Stopwatch timer = logger.isDebugEnabled() ? Stopwatch.createStarted() : null; - final GroupScan newGroupScan =
groupScan.applyFilter(conditionExp,optimizerContext, + AbstractParquetGroupScan newGroupScan = groupScan.applyFilter(conditionExp,
optimizerContext, optimizerContext.getFunctionRegistry(), optimizerContext.getPlannerSettings().getOptions()); if (timer != null) { logger.debug("Took {}
ms to apply filter on parquet row groups. ", timer.elapsed(TimeUnit.MILLISECONDS)); timer.stop(); } - if (newGroupScan == null ) { + // For the case
when newGroupScan wasn't created, the old one may + // fully match the filter for the case when row group pruning did not happen. + if (newGroupScan ==
null) { + if (groupScan.isMatchAllRowGroups()) { + RelNode child = project == null ? scan : project; + // If current row group fully matches filter, +
// but row group pruning did not happen, remove the filter. + if (nonConvertedPredList.size() == 0) { + call.transformTo(child); + } else if
(nonConvertedPredList.size() == predList.size()) { + // None of the predicates participated in filter pushdown. + return; + } else { + // If some of the
predicates weren't used in the filter, creates new filter with them + // on top of current scan. Excludes the case when all predicates weren't used in the filter. +
call.transformTo(filter.copy(filter.getTraitSet(), child, + RexUtil.composeConjunction( + filter.getCluster().getRexBuilder(), + nonConvertedPredList, +
true))); + } + } return; } - RelNode newScan = new ScanPrel(scan.getCluster(), scan.getTraitSet(), newGroupScan, scan.getRowType(), scan.getTable()); +
```

RelNode newNode = new ScanPrel(scan.getCluster(), scan.getTraitSet(), newGroupScan, scan.getRowType(), scan.getTable()); if (project != null) { - newScan = project.copy(project.getTraitSet(), Collections.singletonList(newScan)); + newNode = project.copy(project.getTraitSet(), Collections.singletonList(newNode)); } - if (newGroupScan instanceof AbstractParquetGroupScan) { - RowsMatch matchAll = RowsMatch.ALL; - List<RowGroupInfo> rowGroupInfos = ((AbstractParquetGroupScan) newGroupScan).rowGroupInfos; - for (RowGroupInfo rowGroup : rowGroupInfos) { - if (rowGroup.getRowsMatch() != RowsMatch.ALL) { - matchAll = RowsMatch.SOME; - break; - } - } - if (matchAll == ParquetFilterPredicate.RowsMatch.ALL) { - call.transformTo(newScan); - return; + if (newGroupScan.isMatchAllRowGroups()) { + // creates filter from the expressions which can't be pushed to the scan + if (nonConvertedPredList.size() > 0) { + newNode = filter.copy(filter.getTraitSet(), newNode, + RexUtil.composeConjunction(+ filter.getCluster().getRexBuilder(), + nonConvertedPredList, + true)); } + call.transformTo(newNode); + return; } - final RelNode newFilter = filter.copy(filter.getTraitSet(), Collections.singletonList(newScan)); + final RelNode newFilter = filter.copy(filter.getTraitSet(), Collections.singletonList(newNode)); call.transformTo(newFilter); } } diff --git a/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetRGFilterEvaluator.java b/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetRGFilterEvaluator.java index 281e86569a2..01251498a58 100644 --- a/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetRGFilterEvaluator.java +++ b/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/ParquetRGFilterEvaluator.java @@ -87,8 +87,8 @@ public static RowsMatch matches(LogicalExpression expr, Map<SchemaPath, ColumnSt } Set<LogicalExpression> constantBoundaries = ConstantExpressionIdentifier.getConstantExpressionSet(materializedFilter); - ParquetFilterPredicate parquetPredicate = (ParquetFilterPredicate) ParquetFilterBuilder.buildParquetFilterPredicate(- materializedFilter, constantBoundaries, udfUtilities); + ParquetFilterPredicate parquetPredicate = ParquetFilterBuilder.buildParquetFilterPredicate(+ materializedFilter, constantBoundaries, udfUtilities, true); return matches(parquetPredicate, columnStatisticsMap, rowCount); } diff --git a/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/RowGroupInfo.java b/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/RowGroupInfo.java index 7d2143c1821..1c9ce107cdd 100644 --- a/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/RowGroupInfo.java +++ b/exec/java-exec/src/main/java/org/apache/drill/exec/store/parquet/RowGroupInfo.java @@ -19,7 +19,6 @@ import com.fasterxml.jackson.annotation.JsonCreator; import com.fasterxml.jackson.annotation.JsonProperty; -import org.apache.drill.exec.expr.stat.ParquetFilterPredicate.RowsMatch; import org.apache.drill.exec.store.dfs.ReadEntryFromHDFS; import org.apache.drill.exec.store.dfs.easy.FileWork; import org.apache.drill.exec.store.schedule.CompleteWork; @@ -36,7 +35,6 @@ private List<? extends ColumnMetadata> columns; private long rowCount; // rowCount = -1 indicates to include all rows. private long numRecordsToRead; - private RowsMatch rowsMatch = RowsMatch.SOME; @JsonCreator public RowGroupInfo(@JsonProperty("path") String path, @@ -96,8 +94,4 @@ public long getRowCount() { public void setColumns(List<? extends ColumnMetadata> columns) { this.columns = columns; } - - public RowsMatch getRowsMatch() { return rowsMatch; } - - public void setRowsMatch(RowsMatch rowsMatch) { this.rowsMatch = rowsMatch; } } diff --git a/exec/java-exec/src/test/java/org/apache/drill/exec/store/parquet/TestParquetFilterPushDown.java b/exec/java-exec/src/test/java/org/apache/drill/exec/store/parquet/TestParquetFilterPushDown.java index ea12f40998c..80b06d91679 100644 --- a/exec/java-exec/src/test/java/org/apache/drill/exec/store/parquet/TestParquetFilterPushDown.java +++ b/exec/java-exec/src/test/java/org/apache/drill/exec/store/parquet/TestParquetFilterPushDown.java @@ -70,6 +70,7 @@ public static void initFSAndCreateFragContext() throws Exception { @AfterClass public static void teardown() throws IOException { + fragContext.close(); fs.close(); } @@ -294,6 +295,10 @@ public void testFilterPruning() throws Exception { PlanTestBase.testPlanMatchingPatterns(sql + "a < 1 or a > 1", new String[] {"numRowGroups=3"}); // No filter pruning PlanTestBase.testPlanMatchingPatterns(sql + "a < 1 or a > 2", new String[]{"numRowGroups=2"}, new String[]{"Filter\\("}); //Filter pruning + + // Partial filter pruning + testParquetFilterPruning(sql + "a >=1 and cast(a as varchar) like '%3%'", 1, 2, new String[]{">\\($1, 1\\)"}); + testParquetFilterPruning(sql + "a >=1 and a/3>=1", 2, 2, new String[]{">\\($1, 1\\)"}); } @Test @@ -644,6 +649,15 @@ public void testMinTrueMaxTrue() throws Exception { assertEquals(RowsMatch.ALL, isNotFalse.matches(re)); } + @Test + public void testParquetSingleRowGroupFilterRemoving() throws Exception { + test("create table dfs.tmp.`singleRowGroupTable` as select * from cp.`tpch/nation.parquet`"); + String query = "select * from dfs.tmp.`singleRowGroupTable` where n_nationkey > -1"; + testParquetFilterPruning(query, 25, 1, new String[]{"Filter\\("}); + } + + /// // Some test helper functions. /// ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

27. Merged into Apache master with commit ids [d1a082c..99a3d76|https://github.com/apache/drill/compare/44b990be5c15e1c480725cfb78fcabb40216ebf0..99a3d76551d1a08958c7cd7670df189963fbc943]
28. Verified with Drill version 1.15.0-SNAPSHOT (commit 2dbd60984bef724e6ae1918fa4b31509ca7a986b) Cases checked: - Direct case - Several filters - Different types - Casting in filters - Subqueries A bug was reported during the verification: DRILL-6905.