Item 141
git_comments:

git_commits:

1. **summary:** [SPARK-26709][SQL] OptimizeMetadataOnlyQuery does not handle empty records correctly
   message: [SPARK-26709][SQL] OptimizeMetadataOnlyQuery does not handle empty records correctly ## What changes were proposed in this pull request? When reading from empty tables, the optimization `OptimizeMetadataOnlyQuery` may return wrong results: ``` sql("CREATE TABLE t (col1 INT, p1 INT) USING PARQUET PARTITIONED BY (p1)") sql("INSERT INTO TABLE t PARTITION (p1 = 5) SELECT ID FROM range(1, 1)") sql("SELECT MAX(p1) FROM t") ``` The result is supposed to be `null`. However, with the optimization the result is `5`. The rule is originally ported from https://issues.apache.org/jira/browse/HIVE-1003 in #13494. In Hive, the rule is disabled by default in a later release(https://issues.apache.org/jira/browse/HIVE-15397), due to the same problem. It is hard to completely avoid the correctness issue. Because data sources like Parquet can be metadata-only. Spark can't tell whether it is empty or not without actually reading it. This PR disable the optimization by default. ## How was this patch tested? Unit test Closes #23635 from gengliangwang/optimizeMetadata. Lead-authored-by: Gengliang Wang <gengliang.wang@databricks.com> Co-authored-by: Xiao Li <gatorsmile@gmail.com> Signed-off-by: gatorsmile <gatorsmile@gmail.com> (cherry picked from commit f5b9370da2745a744f8b2f077f1690e0e7035140) Signed-off-by: gatorsmile <gatorsmile@gmail.com>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-15752] [SQL] Optimize metadata only query that has an aggregate whose children are deterministic project or filter operators.
   body: ## What changes were proposed in this pull request? when query only use metadata (example: partition key), it can return results based on metadata without scanning files. Hive did it in HIVE-1003. ## How was this patch tested? add unit tests

github_pulls_comments:

1. **[Test build #59925 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/59925/consoleFull)** for PR 13494 at commit [`2ca2c38`](https://github.com/apache/spark/commit/2ca2c38643d648f48638eb90b2a17b099047ce70). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds no public classes.
2. **[Test build #59929 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/59929/consoleFull)** for PR 13494 at commit [`edea710`](https://github.com/apache/spark/commit/edea710f826477de000913b58568cf9fd4b55814). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds no public classes.
3. **[Test build #59930 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/59930/consoleFull)** for PR 13494 at commit [`8426522`](https://github.com/apache/spark/commit/8426522e0b91376a4922f4059543d7c4fe062ac6). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds no public classes.
4. **[Test build #59940 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/59940/consoleFull)** for PR 13494 at commit [`153293e`](https://github.com/apache/spark/commit/153293ea976f78b0b55d6744a66ecab7e8bedb62). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
5. Can you try to write a design doc on this? Would be great to discuss the reasons why we might want this, the kind of queries that can be answered, corner cases, and how it should be implemented. Thanks.
6. @rxin I have writed a design doc: https://docs.google.com/document/d/1Bmi4-PkTaBQ0HVaGjIqa3eA12toKX52QaiUyhb6WQiM/edit?usp=sharing. Glad to get your comments. Thanks.
7. hi @lianhuiwang , thanks for working on it! The overall idea LGTM, we should elimiante unnecessary file scan if only partition columns are read. However, the current implementation looks not corrected, we should also consider the number of rows. I also took a look at the hive path, it only optimize partition columns used as aggregation keys, where the number of duplicated rows doesn't matter. I think we should either narrow down the scope of this PR and focus on aggregation queries, or spent some more time for a more general design. cc @yhuai @liancheng
8. @cloud-fan Yes, I think what you said is right. as Hive/Prestodb, if queries that did some functions (example: MIN/MAX) or distinct aggregates on partition column and the value of config 'spark.sql.optimizer.metadataOnly' is true, then we can use the metadata-only optimization. I will add a metadataOnly optimizer to optimizer list.Thanks.
9. **[Test build #61161 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61161/consoleFull)** for PR 13494 at commit [`7d7ece0`](https://github.com/apache/spark/commit/7d7ece0fbbe0483f03986b0d49f710ef03d534ba). - This patch **fails to build**. - This patch merges cleanly. - This patch adds no public classes.
10. @cloud-fan Now i have added a extendedHiveOptimizerRules that include MetadataOnly Optimization for Hive Optimizer. Firstly,MetadataOnly Optimization should be in Hive Model because MetastoreRelation only can be used in

Hive now. Secondly, MetadataOnly Optimization should be between Analyzer and RewriteDistinctAggregates. In the future, we can add ParquetConversions/OrcConversions and other optimizations into extendedHiveOptimizerRules.

11. Why is this rule Hive specific?
12. @rxin good point. Because now MetastoreRelation only be defined in Hive now and if we make it using MetadataOnly optimization, like this PR we can use MetadataOnly optimization in Hive Component. if not, we needs divide MetadataOnly optimization into two part, one for common sql, other for HiveQl. I will think more about it and try my best to resolve it. Thanks.
13. **[Test build #61162 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61162/consoleFull)** for PR 13494 at commit [`2e55a9d`](https://github.com/apache/spark/commit/2e55a9df94ef85345f54f2bb3b8c23d06f600f58). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
14. **[Test build #61163 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61163/consoleFull)** for PR 13494 at commit [`b2b6eba`](https://github.com/apache/spark/commit/b2b6eba2ad21675173227ee19c65f6661534e6b6). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
15. **[Test build #61164 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61164/consoleFull)** for PR 13494 at commit [`c5a291e`](https://github.com/apache/spark/commit/c5a291ea6574bd107efe8eb67fde4c48fd1809d0). - This patch passes all tests. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `public class JavaPackage` - `case class StreamingRelationExec(sourceName: String, output: Seq[Attribute]) extends LeafExecNode`
16. @rxin @cloud-fan I have updated this PR for many comments. Can you take a look at? Thanks.
17. **[Test build #61181 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61181/consoleFull)** for PR 13494 at commit [`6404c1f`](https://github.com/apache/spark/commit/6404c1fa2e8e9796574b69b57054dd4791ff6c52). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class MetadataOnlyOptimizer(catalog: SessionCatalog) extends Rule[LogicalPlan]`
18. **[Test build #61185 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61185/consoleFull)** for PR 13494 at commit [`1bb5812`](https://github.com/apache/spark/commit/1bb5812a3af9fffcbc879fba42ccfe66485617bb). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class MetadataOnlyOptimizer(catalog: SessionCatalog) extends Rule[LogicalPlan]`
19. **[Test build #61187 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61187/consoleFull)** for PR 13494 at commit [`7e3729e`](https://github.com/apache/spark/commit/7e3729e407417b58bd5a406e43ab2b1901761699). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds no public classes.
20. **[Test build #61191 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61191/consoleFull)** for PR 13494 at commit [`fbf5d61`](https://github.com/apache/spark/commit/fbf5d615713caab8a6d5aa8093cdb2ef59020966). - This patch passes all tests. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `case class MetadataOnlyOptimizer(`
21. **[Test build #61271 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61271/consoleFull)** for PR 13494 at commit [`3411fd6`](https://github.com/apache/spark/commit/3411fd6e51758d8f556b9cbf3abbc1a7fda952e1). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
22. Hi @lianhuiwang , thanks for working on it, looks pretty good! Some suggestions about the strategy: 1. instead of handling `Distinct`, I think we can expand the optimizable cases in `Aggregate`, to also support `SELECT col FROM tbl GROUP BY col`. 2. instead of traversing the plan tree and collect project list and filter conditions, I think we can replace the table relation with `LogicalRDD` of partition values and keep the `Filter` and `Project` operators.
23. @cloud-fan Thanks for addressing two points. For 1, I have updated for it. For 2, Now at the end it just replace relation plan with LogicalRDD. Thanks.
24. **[Test build #61304 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61304/consoleFull)** for PR 13494 at commit [`aefab7f`](https://github.com/apache/spark/commit/aefab7fa2b2c99f861faf0693fc1e13c2cf5311c). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
25. @cloud-fan Thanks. I have addressed your comments and fixed 'select col from table group by rollup(col)'. add MetadataOnlyOptimizer into end of all optimizations in order to cover all aggregation's cases.
26. **[Test build #61307 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61307/consoleFull)** for PR 13494 at commit [`c5ccdea`](https://github.com/apache/spark/commit/c5ccdea435789ec5cdea1438ace09d91e7726d22). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
27. **[Test build #61308 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61308/consoleFull)** for PR 13494 at commit [`ae6cf9f`](https://github.com/apache/spark/commit/ae6cf9ff909e4c375d4d0f688603f415edff6dca). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
28. **[Test build #61313 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61313/consoleFull)** for PR 13494 at commit

[`159331b`](https://github.com/apache/spark/commit/159331b4cc0583b6414152512d84696c16b895fc). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
29. **[Test build #61373 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61373/consoleFull)** for PR 13494 at commit [`3a1438b`](https://github.com/apache/spark/commit/3a1438ba41c94ed44dbc0dc43c2457509f5e4fcc). - This patch passes all tests. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `trait CatalogRelation extends LogicalPlan`
30. @cloud-fan Thanks. Now i add 'isPartitionColumn' (maybe use other name) to AttributeReference in order to check partition columns in Aggregate.
31. **[Test build #61468 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61468/consoleFull)** for PR 13494 at commit [`c0a7d59`](https://github.com/apache/spark/commit/c0a7d59b014f55b348aa5ed5b089ee87936d2413). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
32. **[Test build #61470 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61470/consoleFull)** for PR 13494 at commit [`a4045ca`](https://github.com/apache/spark/commit/a4045ca3e492d76ae6f7d919cd1130bc9d1a52c5). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
33. **[Test build #61471 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61471/consoleFull)** for PR 13494 at commit [`0a023e7`](https://github.com/apache/spark/commit/0a023e7ce7a20af0b25d320e7fd4455f9e1b3029). - This patch passes all tests. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `trait CatalogRelation`
34. **[Test build #61475 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61475/consoleFull)** for PR 13494 at commit [`a9b38ab`](https://github.com/apache/spark/commit/a9b38abf0972bd956142278379e2e2e546c2b1a1). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
35. **[Test build #61482 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61482/consoleFull)** for PR 13494 at commit [`1bed08d`](https://github.com/apache/spark/commit/1bed08deb2eba2426b83bd5f02dcb0a2ce18afd6). - This patch **fails MiMa tests**. - This patch merges cleanly. - This patch adds no public classes.
36. **[Test build #61481 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61481/consoleFull)** for PR 13494 at commit [`a5ea995`](https://github.com/apache/spark/commit/a5ea995929358ab999f0636d053901a9af84a548). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds no public classes.
37. retest it please.
38. hi @lianhuiwang , we need to propagate partition information bottom up, but adding a `isMetadataColumn` to `NamedExpression` is kind of overkill. I tried to simplify it a little bit, and make it arguably easier to understand. Could you take a look at https://github.com/apache/spark/compare/master...cloud-fan:metadata-only?expand=1 ? thanks!
39. @cloud-fan Thanks. I will look at it.
40. @cloud-fan I have updated with your branch code. Thanks a lot.
41. **[Test build #61609 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61609/consoleFull)** for PR 13494 at commit [`88f7308`](https://github.com/apache/spark/commit/88f7308173829ca2473690a0c409c438d3cd5cf4). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds no public classes.
42. **[Test build #61608 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61608/consoleFull)** for PR 13494 at commit [`41fef2c`](https://github.com/apache/spark/commit/41fef2c40f4929fd26476ecdfa3ee8160394a7d3). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
43. **[Test build #61607 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61607/consoleFull)** for PR 13494 at commit [`a22e962`](https://github.com/apache/spark/commit/a22e9626e6294671e0915822def6eb283a72a643). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
44. **[Test build #61610 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61610/consoleFull)** for PR 13494 at commit [`2568193`](https://github.com/apache/spark/commit/2568193f91b9ae129c19a67bfd514065215840ac). - This patch passes all tests. - This patch merges cleanly. - This patch adds the following public classes _(experimental)_: - `class MetadataOnlyOptimizerSuite extends QueryTest with SharedSQLContext`
45. LGTM exception some naming/testing comments. This is not a small patch and definitely need more reviewers. Let's also add more comments inside the new rule to make it easier to review. cc @yhuai @liancheng to take a look.
46. @cloud-fan Thanks. I have updated with some of your comments. Yes, it is not a small patch and it needs more reviewers. @yhuai @liancheng Could you take a look at this PR? Thanks.
47. **[Test build #61722 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61722/consoleFull)** for PR 13494 at commit [`26a97f4`](https://github.com/apache/spark/commit/26a97f4df483fe324b76c4ee3c1ddec2f3830566). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
48. @lianhuiwang please update the pull request description to include the actual cases that are supported in this pull request.

49. **[Test build #61805 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61805/consoleFull)** for PR 13494 at commit [`4297f9f`](https://github.com/apache/spark/commit/4297f9f7f32fc2ea59bba43569b17da94ed84fce). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
50. @rxin @yhuai @cloud-fan I have addressed your comments. Thanks.
51. **[Test build #61850 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61850/consoleFull)** for PR 13494 at commit [`1a65aa7`](https://github.com/apache/spark/commit/1a65aa7c57e3be349a8823fedc51365e7332981e). - This patch passes all tests. - This patch **does not merge cleanly**. - This patch adds no public classes.
52. **[Test build #61851 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61851/consoleFull)** for PR 13494 at commit [`d5e0df4`](https://github.com/apache/spark/commit/d5e0df4acf3ce6a285b030c4cba42426e146024b). - This patch passes all tests. - This patch **does not merge cleanly**. - This patch adds no public classes.
53. **[Test build #61853 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61853/consoleFull)** for PR 13494 at commit [`9d6dd76`](https://github.com/apache/spark/commit/9d6dd76944eb42411850f8b721a05e263a7db30e). - This patch passes all tests. - This patch **does not merge cleanly**. - This patch adds no public classes.
54. **[Test build #61854 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61854/consoleFull)** for PR 13494 at commit [`9cb01d8`](https://github.com/apache/spark/commit/9cb01d810bc993cae3c98962317026c1e5295ed2). - This patch **fails Spark unit tests**. - This patch **does not merge cleanly**. - This patch adds no public classes.
55. hi @lianhuiwang , I have sent you a PR to improve the document a little bit, do you mind take a look? thanks!
56. @cloud-fan Yes, Thanks, I have merged it.
57. **[Test build #61890 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61890/consoleFull)** for PR 13494 at commit [`88fd3bf`](https://github.com/apache/spark/commit/88fd3bfb13852e1739cef0146209437b417445f3). - This patch passes all tests. - This patch **does not merge cleanly**. - This patch adds no public classes.
58. **[Test build #61891 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61891/consoleFull)** for PR 13494 at commit [`9546b40`](https://github.com/apache/spark/commit/9546b40840e69166c563c491ef6720ccb2f1b2eb). - This patch passes all tests. - This patch **does not merge cleanly**. - This patch adds no public classes.
59. hi @lianhuiwang can you rebase your PR to master? I think it's pretty close!
60. @cloud-fan thanks. I have rebased it to master.
61. **[Test build #61906 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/61906/consoleFull)** for PR 13494 at commit [`67211be`](https://github.com/apache/spark/commit/67211beb80c4d84fb70c6037cc53044f86f094d5). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
62. @hvanhovell can you take a look at this too?
63. @lianhuiwang I left a few smallish comments. Overall LGTM.
64. @hvanhovell I have addressed your comments. Thanks. If I missed something, please tell me.
65. @hvanhovell I have addressed some of your comments. Thanks. Could you look at again?
66. **[Test build #62110 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/62110/consoleFull)** for PR 13494 at commit [`d888c85`](https://github.com/apache/spark/commit/d888c85a1b95904408d436271a40d27095e2eb4d). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
67. @cloud-fan @hvanhovell about getPartitionAttrs() It has a improve place that we can define it in relation node. but now relation node has not this function. how about added in follow-up PRs? Thanks.
68. **[Test build #62137 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/62137/consoleFull)** for PR 13494 at commit [`ff16509`](https://github.com/apache/spark/commit/ff1650987b901825d3828de972fa4434466c951f). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
69. @cloud-fan I have addressed your latest comments. thanks.
70. **[Test build #62156 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/62156/consoleFull)** for PR 13494 at commit [`030776a`](https://github.com/apache/spark/commit/030776ae49484c4e5db7f775344e5e40dff27e9a). - This patch passes all tests. - This patch merges cleanly. - This patch adds no public classes.
71. LGTM - Merging to master. Thanks!
72. Thank you for review and merging. @rxin @hvanhovell @cloud-fan .

github_pulls_reviews:

1. what if this partition has more than one data files?
2. Now in this PR, default of spark.sql.optimizer.metadataOnly is false, So if user needs this feature, he should set spark.sql.optimizer.metadataOnly=true.
3. I think optimizer should never affect the correctness of the query result. If this optimization is too hard to implement with current code base, we should improve the code base first, instead of rushing in a partial implementation.
4. Yes, I rethink more and then i will add a metadataOnly optimizer to optimizer list.Thanks.
5. can we turn it on by default?

6. can we add more comments to explain the condition to enable metadata optimization? e.g. the agg expression must only reference to partition columns, all distinct agg functions, `Max` and `Min`, etc.
7. `MetastoreRelation` extends `CatalogRelation`, I think we can put this rule in sql core instead of hive module.
8. is it safe to keep traversing the plan tree through `Project`?
9. why do we handle `Distinct` specially? It will be rewritten into `Aggregate` and I think we should have a general rule to handle `Aggregate`
10. here is my thoughts about the optimizable cases: First of all, only parition colums are required(which means we need to traverse down the plan tree and find table relation here) 1. aggregate expression is partition columns, e.g. `SELECT col FROM tbl GROUP BY col` 2. aggregate function on partition columns with DISTINCT, e.g. `SELECT count(DISTINCT a) FROM tbl GROUP BY b` 3. aggregate function on partition columns which have same result with or without DISTINCT keyword, e.g. `SELECT sum(a) FROM tbl GROUP BY b`
11. instead of collecting project list and filter conditions, produce a `LogicalRDD` directly, can we just replace the table relation with `LogicalRDD` and still keep these `Filter` and `Project` operators?
12. the `filters` are not guaranteed to be applied, see https://github.com/apache/spark/blob/master/sql/hive/src/main/scala/org/apache/spark/sql/hive/client/HiveShim.scala#L283-L292 We still need to apply the filters
13. ~~for the sake of simplicity, how about we remove this optimization and always apply filters after getting the partition values? We can implement it in follow-ups.~~ Sorry I replied to the wrong place...
14. for the sake of simplicity, how about we remove this optimization and always apply filters after getting the partition values? We can implement it in follow-ups.
15. how about ``` It's used for operators that only need distinct values. Currently only [[Aggregate]] operator which satisfy the following conditions are supported: 1. ..... 2. ..... ```
16. where do we check it only requires partition columns?
17. We need to think about it more carefully, i.e. how can the partition information propagate up from table relation? It's obvious that `Filter` can retain all partition information, but for others, it's not trivial to explain. Since this PR definitely need more people to review, how about we only handle `Filter` for now and improve it later? Then it's easier for other people to review and get this PR in. Thanks!
18. this example is wrong, we can not aggregate on grouping columns, it should be `SELECT count(DISTINCT col1) FROM tbl GROUP BY col2`
19. put these codes inside `withTable` when we create a table in the tests, then our framework will clean up the table automatically. ``` withTable("xxx") { ... } ```
20. This is something we need to discuss, there may be a lot of partition values and using `LocalRelation` may not give enough parallelism here. cc @yhuai @liancheng
21. `Metadata Only` seems not so intuitive, any ideas here? cc @yhuai @liancheng
22. The session is shared among all test suites, so we should drop the table after all tests here, or we may pollute other test suites.
23. the last 2 cases can be added in follow-up PRs :)
24. also use `withSQLConf` here to make sure the optimization is enabled.
25. How about sizeInBytes of LocalRelation's statistics is used to determine the parallelism? Initial code is https://github.com/apache/spark/pull/13979/commits/2ca01f26df7572251136d2c059299f846cf8a3f1.
26. BTW: I find that currently Hive/Presto/Impala have used Metadata Only.
27. Yes, I think it is not difficult.
28. "Optimize Metadata Only Query"
29. ok, Thanks. I will update it.
30. does col2 need to be a partition column?
31. OptimizeMetadataOnlyQuery if you don't add "query" it is not clear to me this actually optimizing a query or optimizing some metadata operation.
32. Yea, it's indicated by the first sentence of the class doc: `When scanning only partition columns`
33. OK I would say something like the following to make it very explicit. ``` This rule optimizes the execution of queries that can be answered by looking only at partition-level metadata. This applies when all the columns scanned are partition columns, and the query has an aggregate operator that satisfies the following conditions: 1. 2. 3. ```
34. we need comment explaining what this function does
35. use named argument, e.g. ``` val partitionData = fsRelation.location.listFiles(filters = Nil) ``` for a while I was wondering what that argument does.
36. this cannot be hit unless there is a bug in Spark right? if that's the case, add a comment saying that.
37. i think `aggFunctions.isEmpty` is not necessary, since forall returns true if aggFunctions is empty?
38. does this work for ``` select col1, col2 from table group by col1 ``` ? It'd be good to handle that too.
39. actually we should probably just rename this function to make it more self evident, e.g. replaceTableScanWithPartitionMetadata
40. we need comment explaining what this does
41. also need to document what the returned tuple means
42. we should also explain what patterns are acceptable, e.g. filter and project with deterministic expressions.
43. it can't pass analysis... `col2` is not grouping column and must be put inside agg functions.
44. a nit: we can save some whitespace and one line ``` unapply(child).flatMap { case (partAttrs, relation) => if (p.references.subsetOf(partAttrs)) Some(p.outputSet, relation) else None } ```
45. same thing here
46. assertMetadataOnlyQuery

47. assertNotMetadataOnlyQuery
48. add a space after collect
49. it'd be great to break this into multiple cases, rather than simply "yes" vs "no".
50. for example, you can have a test case for each of the category you documented in the optimizer classdoc comment, and also have separate test cases for filter, project, etc.
51. why isn't this one supported?
52. ``` select col1, max(col2) from table group by col1 ```
53. because we turn off the flag for this test
54. ah ok - this is getting confusing. we should just have checkWithMetadataOnly test the behavior when the flag is off.
55. Please update the doc to explain what `metadata-only query` means.
56. Do we need to change this file?
57. that's a good point. The new rule is in sql core module and can access the `SQLConf`, we don't need to add the conf in `CatalystConf`
58. yea, as long as `col1` and `col2` are both partition columns
59. I'd name these ... "col1", "col2", "partcol1", "partcol2". to make it very obvious which is partition column and which is not.
60. `Now donot support metadata only optimizer` What this means?
61. This PR do not handle them, They will be added in follow-up PRs. I will delete these two cases because OptimizeMetadataOnlyQuerySuite have included them.
62. why this test is so different from the one in sql core `SQLQuerySuite`?
63. I will make them having same cases. thanks.
64. First/Last?
65. Logic here is the same as the CatalogRelation case. Move this into a separate method?
66. Logic here is the same as the Filter case. Move this into a separate method?
67. There is no need to determine the `partAttrs` again; we can just pass them as an argument.
68. MINOR: all these test have the same structure. We could move this into a function...
69. Yes, we need to handle them. thanks.
70. There is some cases that PartitionedRelation's partAttrs are not same with partAttrs of relation. example: ' select max(c1) from (select partcol1 + 1 as c1 from srcpart where partcol1 = 0) t ' Because we needs to return outputSet of project(eg: c1) in order to check whether Aggregate scan only partitioned attributes. But in this situation, the outputSet of Project are Alias set , not partAttrs of relation.
71. Because there are two lines, I think it is unnecessary for a separate method that causes extra method called.
72. Because there are three lines, I think it is unnecessary for a separate method that causes extra method called.
73. as @rxin addressed before, Dividing this into multiple functions is to have separate test cases for each of the category that has documented in the OptimizeMetadataOnlyQuery.
74. That is not what I mean. I am all for thorough testing, however the structure is the same everywhere. You could generalize this (and improve readability), e.g.: ``` scala def testMetadataOnly(name: String, sql: String*): Unit = { test(name) { withSQLConf(SQLConf.OPTIMIZER_METADATA_ONLY.key -> "true") { sqls.foreach(assertMetadataOnlyQuery(sql(_))) } withSQLConf(SQLConf.OPTIMIZER_METADATA_ONLY.key -> "false") { sqls.foreach(assertNotMetadataOnlyQuery(sql(_))) } } } testMetadataOnly("OptimizeMetadataOnlyQuery test: aggregate expression is partition columns", "select partcol1 from srcpart group by partcol1", "select partcol2 from srcpart where partcol1 = 0 group by partcol2") ```
75. Two lines is IMO not a valid argument for not factoring this out. The code here is not entirely trivial, and does take the reviewer a little bit of time to understand what is going on.
76. Get it, thanks. I will update it. Thanks.
77. Nit: Style. ``` def getPartitionAttrs(partitionColumnNames: Seq[String], relation: LogicalPlan): Seq[Attribute] = { ... ``` While you are at it. Change the return type to AttributeSet.
78. When is this one used?
79. NVM - I am blind
80. does `getPartitionAttrs` need to be a method in `PartitionedRelation`? I think it can just be a private method in parent class.
81. classes under `execution` package are considered as private, we don't need to mark them as private explicitly.
82. thanks. Because object PartitionedRelation also use getPartitionAttrs, Now i just define it in PartitionedRelation. If it define a private method in class OptimizeMetadataOnlyQuery, there are two same getPartitionAttrs() functions in PartitionedRelation and OptimizeMetadataOnlyQuery. How about define two same getPartitionAttrs() functions? or has another way?
83. Get it. Thanks.
84. Get it. Thanks.
85. @cloud-fan I will define two functions for getPartitionAttrs(). In the future, I think we can put getPartitionAttrs() into relation plan. If i has some problem, please tell me. thanks.
86. IIRC, inner class can access private member of outer class, we don't need to duplicate the method in inner class.
87. I think we can remove the prefix: `OptimizeMetadataOnlyQuery test`. The test report will print the name of this test suite for these tests.
88. Yes, thanks.
89. Sure. thanks.

jira_issues:

1. **summary:** optimize metadata only queries
   description: Queries like: select max(ds) from T where ds is a partitioning column should be optimized.

jira_issues_comments:

1. https://reviews.apache.org/r/1962/
2. cleaning up the patch
3. I have finished my internship at Facebook. If you want to contact me you can reach me at Marcin Kurczych <marcin.kurczych@gmail.com>. Thanks!
4. will take a look.
5. will add more tests
6. added more tests
7. @Namit, Will second query on HIVE-2119 will also get optimized with this?
8. @Ashutosh, it should - i haven't tried it yet. Let me confirm tomorrow
9. @Ashutosh, that query is not optimized - Let us keep HIVE-2119 open and work on that as a followup.
10. 1) move the optimizer to physical optimizer. 2) after 1), changes in GenMapRedUtils also need to be moved. 3) add a testcase which uses CombineHiveInputFormat. 4) also test virtual_column.q nitpicks: 1) getCategory() in NullStructSerDe's getObjectInspector should return Struct 2) remove import of "HiveNullValueSequenceFileOutputFormat" and "OneNullRowInputFormat" from SemanticAnalyzer.java
11. @Ashutosh, moving to physical optimizer should also fix the query in HIVE-2119
12. njain requested code review of "HIVE-1003 [jira] optimize metadata only queries". Reviewers: JIRA testing Queries like: select max(ds) from T where ds is a partitioning column should be optimized. TEST PLAN EMPTY REVISION DETAIL https://reviews.facebook.net/D105 AFFECTED FILES ql/src/java/org/apache/hadoop/hive/ql/parse/SemanticAnalyzer.java MANAGE HERALD DIFFERENTIAL RULES https://reviews.facebook.net/herald/view/differential/ WHY DID I GET THIS EMAIL? https://reviews.facebook.net/herald/transcript/219/ Tip: use the X-Herald-Rules header to filter Herald messages in your client.
13. njain has commented on the revision "HIVE-1003 [jira] optimize metadata only queries". testing integration REVISION DETAIL https://reviews.facebook.net/D105
14. addressed comments
15. looks good to me, running tests
16. committed, thanks Marcin and Namit!
17. Integrated in Hive-trunk-h0.21 #1048 (See [https://builds.apache.org/job/Hive-trunk-h0.21/1048/]) HIVE-1003: optimize metadata only queries (Marcin Kurczych, Namit Jain via He Yongqiang) heyongqiang : http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1195577 Files : * /hive/trunk/common/src/java/org/apache/hadoop/hive/conf/HiveConf.java * /hive/trunk/jdbc/src/java/org/apache/hadoop/hive/jdbc/HiveDatabaseMetaData.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/exec/ExecDriver.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/exec/MapRedTask.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/exec/MapredLocalTask.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/exec/Task.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/io/OneNullRowInputFormat.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/optimizer/physical/MetadataOnlyOptimizer.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/optimizer/physical/PhysicalOptimizer.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/plan/GroupByDesc.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/udf/UDFType.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/udf/generic/GenericUDAFMax.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/udf/generic/GenericUDAFMin.java * /hive/trunk/ql/src/test/queries/clientpositive/metadataonly1.q * /hive/trunk/ql/src/test/results/clientpositive/metadataonly1.q.out * /hive/trunk/serde/src/java/org/apache/hadoop/hive/serde2/NullStructSerDe.java
18. njain updated the revision "HIVE-1003 [jira] optimize metadata only queries". Reviewers: JIRA REVISION DETAIL https://reviews.facebook.net/D105 AFFECTED FILES ql/src/test/results/clientpositive/join0.q.out ql/src/test/queries/clientpositive/groupby2.q ql/src/test/queries/clientpositive/join0.q ql/src/java/org/apache/hadoop/hive/ql/exec/ExplainTask.java ql/src/java/org/apache/hadoop/hive/ql/plan/ExplainWork.java ql/src/java/org/apache/hadoop/hive/ql/parse/Hive.g ql/src/java/org/apache/hadoop/hive/ql/parse/ExplainSemanticAnalyzer.java
19. njain updated the revision "HIVE-1003 [jira] optimize metadata only queries". Reviewers: JIRA HIVE-2618 REVISION DETAIL https://reviews.facebook.net/D105 AFFECTED FILES ql/src/test/results/clientpositive/alter_table_serde.q.out ql/src/test/results/clientpositive/exim_04_evolved_parts.q.out ql/src/test/results/clientpositive/partition_schema1.q.out ql/src/test/queries/clientpositive/partition_schema1.q ql/src/java/org/apache/hadoop/hive/ql/metadata/MetaDataFormatUtils.java ql/src/java/org/apache/hadoop/hive/ql/exec/DDLTask.java
20. heyongqiang has accepted the revision "HIVE-1003 [jira] optimize metadata only queries". tests passed REVISION DETAIL https://reviews.facebook.net/D105
21. njain has committed the revision "HIVE-1003 [jira] optimize metadata only queries". REVISION DETAIL https://reviews.facebook.net/D105 COMMIT https://reviews.facebook.net/rHIVE1211767
22. Integrated in Hive-trunk-h0.23.0 #7 (See [https://builds.apache.org/job/Hive-trunk-h0.23.0/7/]) HIVE-1003 [jira] optimize metadata only queries (Namit Jain via Yongqiang He) Summary: testing Queries like: select max(ds) from T where ds is a partitioning column should be optimized. Test Plan: EMPTY Reviewers: JIRA, heyongqiang Reviewed By: heyongqiang CC: njain, heyongqiang Differential Revision: 105 heyongqiang : http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1211767 Files : * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/exec/DDLTask.java *

/hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/metadata/MetaDataFormatUtils.java *
/hive/trunk/ql/src/test/queries/clientpositive/partition_schema1.q *
/hive/trunk/ql/src/test/results/clientpositive/alter_table_serde.q.out *
/hive/trunk/ql/src/test/results/clientpositive/exim_04_evolved_parts.q.out *
/hive/trunk/ql/src/test/results/clientpositive/partition_schema1.q.out

23. Integrated in Hive-trunk-h0.21 #1131 (See [https://builds.apache.org/job/Hive-trunk-h0.21/1131/]) HIVE-1003 [jira] optimize metadata only queries (Namit Jain via Yongqiang He) Summary: testing Queries like: select max(ds) from T where ds is a partitioning column should be optimized. Test Plan: EMPTY Reviewers: JIRA, heyongqiang Reviewed By: heyongqiang CC: njain, heyongqiang Differential Revision: 105 heyongqiang : http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1211767 Files : * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/exec/DDLTask.java * /hive/trunk/ql/src/java/org/apache/hadoop/hive/ql/metadata/MetaDataFormatUtils.java * /hive/trunk/ql/src/test/queries/clientpositive/partition_schema1.q * /hive/trunk/ql/src/test/results/clientpositive/alter_table_serde.q.out * /hive/trunk/ql/src/test/results/clientpositive/exim_04_evolved_parts.q.out * /hive/trunk/ql/src/test/results/clientpositive/partition_schema1.q.out
24. Doc note: This adds *hive.optimize.metadataonly* to HiveConf.java with a default value of true in release 0.8.0. HIVE-15397 changes the default to false in release 2.2.0. When it is documented in the wiki, this link will work: * [Configuration Properties -- hive.optimize.metadataonly | https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties#ConfigurationProperties-hive.optimize.metadataonly]