Item 103
git_comments:

git_commits:

1. **summary:** [SPARK-15616][SQL] Add optimizer rule PruneHiveTablePartitions
   message: [SPARK-15616][SQL] Add optimizer rule PruneHiveTablePartitions ### What changes were proposed in this pull request? Add optimizer rule PruneHiveTablePartitions pruning hive table partitions based on filters on partition columns. Doing so, the total size of pruned partitions may be small enough for broadcast join in JoinSelection strategy. ### Why are the changes needed? In JoinSelection strategy, spark use the "plan.stats.sizeInBytes" to decide whether the plan is suitable for broadcast join. Currently, "plan.stats.sizeInBytes" does not take "pruned partitions" into account, so it may miss some broadcast join and take sort-merge join instead, which will definitely impact join performance. This PR aim at taking "pruned partitions" into account for hive table in "plan.stats.sizeInBytes" and then improve performance by using broadcast join if possible. ### Does this PR introduce any user-facing change? no ### How was this patch tested? Added unit tests. This is based on #25919, credits should go to lianhuiwang and advancedxy. Closes #26805 from fuwhu/SPARK-15616. Authored-by: fuwhu <bestwwg@163.com> Signed-off-by: Wenchen Fan <wenchen@databricks.com>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-15616][SQL] Hive table supports partition pruning in JoinSelection
   body: ### What changes were proposed in this pull request? A new optimizer strategy called `PruneHiveTablePartitions` is added, which calculates table size as the total size of pruned partitions. Thus, Spark planner can pick up `BroadcastJoin` if the size of pruned partitions is under broadcast join threshold. ### Why are the changes needed? This is a performance improvement. ### Does this PR introduce any user-facing change? No. ### How was this patch tested? Added unit tests. This is based on #18193, credits should go to @lianhuiwang.

github_pulls_comments:

1. cc @cloud-fan.
2. ok to test
3. add to whitelist
4. **[Test build #111297 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/111297/testReport)** for PR 25919 at commit [`ffb7168`](https://github.com/apache/spark/commit/ffb71684bd7c2b48ab10f0acbec3df89b9b3f6b6). * This patch **fails Spark unit tests**. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `case class PruneHiveTablePartitions(`
5. @cloud-fan Now pruned partitions are cached in HiveTableRelation, what do you think about current approach ?
6. **[Test build #112331 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/112331/testReport)** for PR 25919 at commit [`c054d22`](https://github.com/apache/spark/commit/c054d22dda61883737c217459e36797f0bc796ba). * This patch **fails to build**. * This patch **does not merge cleanly**. * This patch adds no public classes.
7. **[Test build #112332 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/112332/testReport)** for PR 25919 at commit [`e744da5`](https://github.com/apache/spark/commit/e744da59aad15e8d1f5c7582fab195ddfde0a18c). * This patch **fails to build**. * This patch merges cleanly. * This patch adds no public classes.
8. **[Test build #112334 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/112334/testReport)** for PR 25919 at commit [`12e1dc5`]

(https://github.com/apache/spark/commit/12e1dc56db32ee4c939e8ad8260d484a30c7b3f8). * This patch **fails to build**. * This patch merges cleanly. * This patch adds no public classes.
9. **[Test build #112337 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/112337/testReport)** for PR 25919 at commit [`b334e99`] (https://github.com/apache/spark/commit/b334e9926cc810f73e2c41934021ef7b5ffe7128). * This patch **fails Spark unit tests**. * This patch merges cleanly. * This patch adds no public classes.
10. **[Test build #112356 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/112356/testReport)** for PR 25919 at commit [`8d615f7`] (https://github.com/apache/spark/commit/8d615f73fccd47f3690af4d4806f120e5c058d9e). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
11. **[Test build #112679 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/112679/testReport)** for PR 25919 at commit [`86a0d9c`] (https://github.com/apache/spark/commit/86a0d9c489501e1e7b42351154973f21a838763b). * This patch **fails Scala style tests**. * This patch merges cleanly. * This patch adds no public classes.
12. **[Test build #112707 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/112707/testReport)** for PR 25919 at commit [`ecfbe4d`] (https://github.com/apache/spark/commit/ecfbe4d975aeee1da38240753c99a6927a8aa690). * This patch **fails due to an unknown error code, -9**. * This patch merges cleanly. * This patch adds no public classes.
13. retest it please
14. Gently ping @cloud-fan
15. retest this please
16. still `WIP`?
17. **[Test build #114331 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/114331/testReport)** for PR 25919 at commit [`ecfbe4d`] (https://github.com/apache/spark/commit/ecfbe4d975aeee1da38240753c99a6927a8aa690). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
18. > still `WIP`? I think it's ready for review.
19. closed in favor of #26805

github_pulls_reviews:

1. Why we need to check `conf.fallBackToHdfsForStatsEnabled`?
2. so we list partitions here just to get statistic, and we will list partitions again at runtime?
3. We should only get size from HDFS if `conf.fallBackToHdfsForStatsEnabled`? Since it could be a time-consuming operation. Though, this condition should probably be pushed down to before the `CommandUtils.calculateLocationSize` call
4. Currently, yes. And I didn't find an easy way to avoid the list partitions operation.
5. Then this may make the performance worse. How about we keep the listed partitions in `HiveTableRelation`?
6. > How about we keep the listed partitions in HiveTableRelation? This is a good one. However, we may have to add two fields: `pruningFilters: Seq[Expression]` and `prunedPartitions: Seq[CatalogTablePartition]`, and I believe they are complicating the `HiveTableRelation`. Another things is that `HiveTableRelation` might be copied multiple times, we may lost the `prunedPartitions` field by accident if not copied carefully. If that's not the problem, storing listed partitions in `HiveTableRelation` is a good choice. WDYT @cloud-fan?
7. How can we distinguish 0 partitions after pruning, and not being partition pruned?
8. can we follow `PruneFileSourcePartitions`? I think we should also support `Filter(Project(HiveScan))`
9. What are we doing here?
10. Why do we need to keep `pruningPredicates`? IIUC the approach should be very simply: 1. this rule only changes `HiveTableRelation` to hold an optional partition list. 2. the `HiveTableScanExec` will get the partition list from `HiveTableRelation` or call `listPartitionsByFilter`.
11. Will do.

12. We have another field called `normalizedFilters`, when it's empty(Nil), then the `prunedPartitions` are not pruned, otherwise it could be 0 partitions after pruning when `prunedPartitions` = Nil
13. Only under exactly matched pruning filters, we can simply get partitions from `HiveTableRelation`
14. Due to [SPARK-24085](https://issues.apache.org/jira/browse/SPARK-24085), the pruningPredicates(we eliminate the subquery) could be different than the filters passed to `HiveTableScan`. So I keep the pruningPredicates, and only retrieves the `prunedPartitions` when `HiveTableScanExec`'s `pruningPartitionPredict` matches exactly with `HiveTableRelation`'s `normalizedFilters`. The simplified solution occurred to me first, then I thought the filters could be different for some reason, and SPARK-24085 is an example, hence the proposed solution here.
15. @cloud-fan @maropu @advancedxy Since the rawPartitions are called by "prunePartitions(rawPartitions)" in doExecute method, it seems prunePartitions will filter out all irrelevant partitions using "boundPruningPred". Then why we still need to call listpartitionsByFilter here ? Could you please help me understand this ? thanks a lot in advance.
16. It skips all subqueries instead of scalar subqueries.
17. Per the doc of the conf "spark.sql.statistics.fallBackToHdfs", it is only for non-partitioned hive table : "This flag is effective only for non-partitioned Hive tables."

jira_issues:

jira_issues_comments: