Item 117
git_comments:

git_commits:

1. **summary:** [SPARK-5852][SQL]Fail to convert a newly created empty metastore parquet table to a data source parquet table.
   message: [SPARK-5852][SQL]Fail to convert a newly created empty metastore parquet table to a data source parquet table. The problem is that after we create an empty hive metastore parquet table (e.g. `CREATE TABLE test (a int) STORED AS PARQUET`), Hive will create an empty dir for us, which cause our data source `ParquetRelation2` fail to get the schema of the table. See JIRA for the case to reproduce the bug and the exception. This PR is based on #4562 from chenghao-intel. JIRA: https://issues.apache.org/jira/browse/SPARK-5852 Author: Yin Huai <yhuai@databricks.com> Author: Cheng Hao <hao.cheng@intel.com> Closes #4655 from yhuai/CTASParquet and squashes the following commits: b8b3450 [Yin Huai] Update tests. 2ac94f7 [Yin Huai] Update tests. 3db3d20 [Yin Huai] Minor update. d7e2308 [Yin Huai] Revert changes in HiveMetastoreCatalog.scala. 36978d1 [Cheng Hao] Update the code as feedback a04930b [Cheng Hao] fix bug of scan an empty parquet based table 442ffe0 [Cheng Hao] passdown the schema for Parquet File in HiveContext

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-5852] [SQL] Passdown the schema for Parquet File in HiveContext
   body: It's not allowed to be the empty directory for parquet, for example, it will failed when query the following ``` CREATE TABLE parquet_test (id int, str string) STORED AS PARQUET; SELECT * FROM parquet_test; ``` It throws exception like: ``` java.lang.UnsupportedOperationException: empty.reduceLeft at scala.collection.TraversableOnce$class.reduceLeft(TraversableOnce.scala:167) at scala.collection.mutable.ArrayBuffer.scala$collection$IndexedSeqOptimized$$super$reduceLeft(ArrayBuffer.scala:47) at scala.collection.IndexedSeqOptimized$class.reduceLeft(IndexedSeqOptimized.scala:68) at scala.collection.mutable.ArrayBuffer.reduceLeft(ArrayBuffer.scala:47) at scala.collection.TraversableOnce$class.reduce(TraversableOnce.scala:195) at scala.collection.AbstractTraversable.reduce(Traversable.scala:105) at org.apache.spark.sql.parquet.ParquetRelation2$.readSchema(newParquet.scala:633) at org.apache.spark.sql.parquet.ParquetRelation2$MetadataCache.org$apache$spark$sql$parquet$ParquetRelation2$MetadataCache$$readSchema(newParquet.scala:3 at org.apache.spark.sql.parquet.ParquetRelation2$MetadataCache$$anonfun$refresh$8.apply(newParquet.scala:290) at org.apache.spark.sql.parquet.ParquetRelation2$MetadataCache$$anonfun$refresh$8.apply(newParquet.scala:290) at scala.Option.getOrElse(Option.scala:120) at org.apache.spark.sql.parquet.ParquetRelation2$MetadataCache.refresh(newParquet.scala:290) at org.apache.spark.sql.parquet.ParquetRelation2.<init>(newParquet.scala:354) at org.apache.spark.sql.hive.HiveMetastoreCatalog.org$apache$sparksqlhive$HiveMetastoreCatalog$$convertToParquetRelation(HiveMetastoreCatalog.scala:218) at org.apache.spark.sql.hive.HiveMetastoreCatalog$ParquetConversions$$anonfun$apply$4.apply(HiveMetastoreCatalog.scala:446) at org.apache.spark.sql.hive.HiveMetastoreCatalog$ParquetConversions$$anonfun$apply$4.apply(HiveMetastoreCatalog.scala:445) at scala.collection.IndexedSeqOptimized$class.foldl(IndexedSeqOptimized.scala:51) at scala.collection.IndexedSeqOptimized$class.foldLeft(IndexedSeqOptimized.scala:60) at scala.collection.mutable.ArrayBuffer.foldLeft(ArrayBuffer.scala:47) at org.apache.spark.sql.hive.HiveMetastoreCatalog$ParquetConversions$.apply(HiveMetastoreCatalog.scala:445) at org.apache.spark.sql.hive.HiveMetastoreCatalog$ParquetConversions$.apply(HiveMetastoreCatalog.scala:422) at org.apache.spark.sql.catalyst.rules.RuleExecutor$$anonfun$apply$1$$anonfun$apply$2.apply(RuleExecutor.scala:61) at org.apache.spark.sql.catalyst.rules.RuleExecutor$$anonfun$apply$1$$anonfun$apply$2.apply(RuleExecutor.scala:59) at scala.collection.LinearSeqOptimized$class.foldLeft(LinearSeqOptimized.scala:111) at scala.collection.immutable.List.foldLeft(List.scala:84) at org.apache.spark.sql.catalyst.rules.RuleExecutor$$anonfun$apply$1.apply(RuleExecutor.scala:59) at org.apache.spark.sql.catalyst.rules.RuleExecutor$$anonfun$apply$1.apply(RuleExecutor.scala:51) at scala.collection.immutable.List.foreach(List.scala:318) at org.apache.spark.sql.catalyst.rules.RuleExecutor.apply(RuleExecutor.scala:51) at org.apache.spark.sql.SQLContext$QueryExecution.analyzed$lzycompute(SQLContext.scala:917) at org.apache.spark.sql.SQLContext$QueryExecution.analyzed(SQLContext.scala:917) at org.apache.spark.sql.DataFrameImpl.<init>(DataFrameImpl.scala:61) at org.apache.spark.sql.DataFrame$.apply(DataFrame.scala:35) at org.apache.spark.sql.hive.HiveContext.sql(HiveContext.scala:77) at org.apache.spark.sql.hive.thriftserver.AbstractSparkSQLDriver.run(AbstractSparkSQLDriver.scala:57) at org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver.processCmd(SparkSQLCLIDriver.scala:275) at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:423) at org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver$.main(SparkSQLCLIDriver.scala:211) at org.apache.spark.sql.hive.thriftserver.SparkSQLCLIDriver.main(SparkSQLCLIDriver.scala) ```

github_pulls_comments:

1. [Test build #27344 has started](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27344/consoleFull) for PR 4562 at commit [`33867c0`](https://github.com/apache/spark/commit/33867c09248b22cca8456268af176f17206a4b74). - This patch merges cleanly.
2. [Test build #27344 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27344/consoleFull) for PR 4562 at commit [`33867c0`](https://github.com/apache/spark/commit/33867c09248b22cca8456268af176f17206a4b74). - This patch **fails Spark unit tests**. - This patch merges cleanly. - This patch adds no public classes.
3. Test FAILed. Refer to this link for build results (access rights to CI server needed): https://amplab.cs.berkeley.edu/jenkins//job/SparkPullRequestBuilder/27344/ Test FAILed.
4. [Test build #27414 has started](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27414/consoleFull) for PR 4562 at commit [`cbb5460`](https://github.com/apache/spark/commit/cbb5460e5fb4565f2d39e0ce71c00899ec7e8a7e). - This patch merges cleanly.
5. [Test build #27414 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27414/consoleFull) for PR 4562 at commit [`cbb5460`](https://github.com/apache/spark/commit/cbb5460e5fb4565f2d39e0ce71c00899ec7e8a7e). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
6. Test PASSed. Refer to this link for build results (access rights to CI server needed): https://amplab.cs.berkeley.edu/jenkins//job/SparkPullRequestBuilder/27414/ Test PASSed.
7. @liancheng
8. [Test build #27607 has started](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27607/consoleFull) for PR 4562 at commit [`a04930b`](https://github.com/apache/spark/commit/a04930badb291e55ba4e6ba79ce781a89f827932). - This patch merges cleanly.
9. [Test build #27607 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27607/consoleFull) for PR 4562 at commit [`a04930b`](https://github.com/apache/spark/commit/a04930badb291e55ba4e6ba79ce781a89f827932). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
10. Test PASSed. Refer to this link for build results (access rights to CI server needed): https://amplab.cs.berkeley.edu/jenkins//job/SparkPullRequestBuilder/27607/ Test PASSed.
11. [Test build #27615 has started](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27615/consoleFull) for PR 4562 at commit [`36978d1`](https://github.com/apache/spark/commit/36978d1835ab6e0266ad3787b33056b573fd59e8). - This patch merges cleanly.
12. [Test build #27615 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/27615/consoleFull) for PR 4562 at commit [`36978d1`](https://github.com/apache/spark/commit/36978d1835ab6e0266ad3787b33056b573fd59e8). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
13. Test PASSed. Refer to this link for build results (access rights to CI server needed): https://amplab.cs.berkeley.edu/jenkins//job/SparkPullRequestBuilder/27615/ Test PASSed.
14. Hey @chenghao-intel @yhuai, sorry I didn't notice this PR earlier, and I believe this issue has been fixed in #4563 ([here](https://github.com/apache/spark/pull/4563/files#diff-c69b9e667e93b7e4693812cc72abb65fR245)).
15. @chenghao-intel can you close it? It is has been fixed by #4655.

16. Thank you @yhuai , I am closing this PR. :)

github_pulls_reviews:

1. I think we cannot do it. See https://github.com/apache/spark/blob/master/sql/hive/src/main/scala/org/apache/spark/sql/hive/HiveMetastoreCatalog.scala#L194
2. How about this ``` parquetSchema = { if (maybeSchema.isDefined) { maybeSchema.get } else { (readSchema(), maybeMetastoreSchema) match { case (Some(dataSchema), _) => dataSchema case (None, Some(metastoreSchema)) => metastoreSchema case (None, None) => throw new SparkException("Failed to get the schema.") } } } ``` We first check if maybeSchema is defined. If not, we read the schema from existing data. If existing data does not exist, we are dealing with a newly created empty table and we will use maybeMetastoreSchema defined in the options.
3. How about using `None` instead of `null`?
4. Also add a test for `CREATE TABLE ... STORED AS PARQUET AS ...`?
5. OK, we can leave this file unchanged.
6. `STORED AS PARQUET` is supported since Hive 0.13, the unit test may failed in Hive 0.12 if we do that.
7. Also, seems we do not need `try ... catch` at here.
8. Yeah, I was trying that also, but seems using `null` is more simple, as `Option` requires some more value extracting code.
9. How about we use `if (HiveShim.version =="0.13.1")` to check the Hive version like what we did in https://github.com/apache/spark/commit/e0490e271d078aa55d7c7583e2ba80337ed1b0c4.
10. After reading the source code, I am wondering if the `maybeMetastoreSchema` is redundant, and it probably should be always converted into `maybeSchema` when creating the `ParquetRelation2` instance?
11. All right. Instead of putting a large code block in `Option`, how about use a temporary `val` and then use `Option` at the end of this method.
12. Based on Cheng's comment at https://github.com/apache/spark/blob/master/sql/hive/src/main/scala/org/apache/spark/sql/hive/HiveMetastoreCatalog.scala#L194, I think that it is better to keep `maybeMetastoreSchema` and we just fix the bug for now.
13. Oh, i thought `STORED AS PARQUERT AS ..` is just the syntactic sugar. Unfortunately, all of the test suite are implemented in the sub project `sql`, but the `HiveShim` is in the subproject `hive` with `hive` package accessing visibility. Let's put this test in another PR?
14. Yeah, evil case insensitivity...

jira_issues:

jira_issues_comments: