Item 274
**git_comments:**

1. if UPDATE_CACHE_FREQUENCY is set, always let the server set timestamps

**git_commits:**

1. **summary:** PHOENIX-3907 Use LATEST_TIMESTAMP when UPDATE_CACHE_FREQUENCY is not zero
   **message:** PHOENIX-3907 Use LATEST_TIMESTAMP when UPDATE_CACHE_FREQUENCY is not zero

**github_issues:**

**github_issues_comments:**

**github_pulls:**

**github_pulls_comments:**

**github_pulls_reviews:**

**jira_issues:**

1. **summary:** Use LATEST_TIMESTAMP when UPDATE_CACHE_FREQUENCY is not zero
   **description:** For non transactional tables, currently with UPDATE_CACHE_FREQUENCY, we'll use LATEST_TIMESTAMP *most* of the time, until the cached entity expires, in which case we'll use the server timestamp. This seems a bit strange and inconsistent. Instead (for non transactional tables), we should always use LATEST_TIMESTAMP if UPDATE_CACHE_FREQUENCY is non zero, with the exception of the corner case for UPSERT SELECT and DELETE where the same table is being read and written to (see changes to FromCompiler for PHOENIX-3823).

**jira_issues_comments:**

1. [~tdsilva] - this would be a nice one to get in since the JIRAs associated with PHOENIX-3819 have been checked in. WDYT? Does this change make sense to you?
2. [~jamestaylor] You are referring to setting the MetaDataMutationResult mutation time to QueryConstants.UNSET_TIMESTAMP when update cache frequency is set and the cached entity hasn't expired right ? Later on when we create mutations if the timestamp is set to UNSET_TIMESTAMP we use the LATEST_TIMESTAMP. {code} // Do not make rpc to getTable if // 1. table is a system table // 2. table was already resolved as of that timestamp if (table != null && !alwaysHitServer && (systemTable || resolvedTimestamp == tableResolvedTimestamp || connection.getMetaDataCache().getAge(tableRef) < table.getUpdateCacheFrequency())) { return new MetaDataMutationResult(MutationCode.TABLE_ALREADY_EXISTS, QueryConstants.UNSET_TIMESTAMP, table); } {code} If the cached entry expired the MetaDataMutationResult mutation time is set to the server timestamp (same as the behavior of tables without UPDATE_CACHE_FREQUENCY set). I think it makes sense to be consistent and always set the MetaDataMutationResult mutation time to QueryConstants.UNSET_TIMESTAMP if UPDATE_CACHE_FREQUENCY is set (except for the UPSERT SELECT/ DELETE corner case when the alwaysHitServer flag is set to true).
3. Would you have some spare cycles to take this one, [~tdsilva]? If it's not too difficult, we can target it for 4.11.0.
4. Sure I can pick this one up.
5. [~jamestaylor] Can you please review? Thanks, Thomas
6. Thanks for the patch, [~tdsilva]. One possible additional consideration is when a query uses CURRENT_DATE(). We typically piggyback on the timestamp we get back from the server as our current date so that we don't have to make another RPC. Can you make sure that we preserve this optimization? You might need to have a member variable in StatementContext that preserves the current date and then in addition, do what you're doing to have the result timestamp always be unset.
7. **body:** {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12871300/PHOENIX-3907.patch against master branch at commit bacaba18bd17d54ff6b24fb5e59c6ab0a6ce62f1. ATTACHMENT ID: 12871300 {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:red}-1 tests included{color}. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:red}-1 javadoc{color}. The javadoc tool appears to have generated 47 warning messages. {color:red}-1 release audit{color}. The applied patch generated 5 release audit warnings (more than the master's current 0 warnings). {color:green}+1 lineLengths{color}. The patch does not introduce lines longer than 100 {color:red}-1 core tests{color}. The patch failed these unit tests: ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.UpgradeIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.SortMergeJoinIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.index.MutableIndexFailureIT Test results: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1000//testReport/ Release audit warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1000//artifact/patchprocess/patchReleaseAuditWarnings.txt Javadoc warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1000//artifact/patchprocess/patchJavadocWarnings.txt Console output: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1000//console This message is automatically generated.
   **label:** code-design
8. [~jamestaylor] StatementContext.getCurrentTime() sets the current time based on the current table reference. If I modify MetadataClient.updateCache to set the mutation result time to UNSET_TIMESTAMP, then we lose the current date optimization that prevents the rpc. I don't see an easy way to get the statement context in updateCache. Instead of modifying

update cache, should I just modify MutationState.validate to set the server time stamp to UNSET_TIMESTAMP if UPDATE_CACHE_FREQUENCY is set ? This will then be used to generate the mutations. {code} - serverTimeStamp = timestamp; + // if UPDATE_CACHE_FREQUENCY is set the mutation result time is UNSET_TIMESTAMP *except* when the cached table expires + // so set the server time stamp to UNSET_TIMESTAMP to be consisent + serverTimeStamp = table.getUpdateCacheFrequency()!=0 ? QueryConstants.UNSET_TIMESTAMP : timestamp; {code}

9. That seems reasonable, [~tdsilva]. Any holes with the upsert select or delete case where we need the time stamp set? Or if SCN is set? Would another option be to add a currentTime member variable to TableRef and capture the timestamp from the result there? I think you could the PTable there to check the updateCacheFrequency value.
10. [~jamestaylor] Thanks for the review, I have uploaded a modified patch.
11. +1. Thanks, [~tdsilva]!
12. **body:** {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12871716/PHOENIX-3907-v2.patch against master branch at commit fbe6ee22d8a6bafa325a56ee7282fa77253b7e07. ATTACHMENT ID: 12871716 {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:red}-1 tests included{color}. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:red}-1 javadoc{color}. The javadoc tool appears to have generated 50 warning messages. {color:red}-1 release audit{color}. The applied patch generated 5 release audit warnings (more than the master's current 0 warnings). {color:green}+1 lineLengths{color}. The patch does not introduce lines longer than 100 {color:red}-1 core tests{color}. The patch failed these unit tests: ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.MutableQueryIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.UpdateCacheAcrossDifferentClientsIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.index.MutableIndexFailureIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.AlterTableWithViewsIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.ViewIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.rpc.UpdateCacheIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.CreateTableIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.AppendOnlySchemaIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.UpgradeIT {color:red}-1 core zombie tests{color}. There are 3 zombie test(s): at org.apache.hadoop.hbase.client.TestAdmin2.testGetRegion(TestAdmin2.java:730) at org.apache.hadoop.hbase.namespace.TestNamespaceAuditor.testRestoreSnapshotQuotaExceed(TestNamespaceAuditor.java:838) Test results: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1014//testReport/ Release audit warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1014//artifact/patchprocess/patchReleaseAuditWarnings.txt Javadoc warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1014//artifact/patchprocess/patchJavadocWarnings.txt Console output: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1014//console This message is automatically generated.
    **label:** code-design
13. [~jamestaylor] I have attached a v3 patch to fix test failures.
14. **body:** {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12871742/PHOENIX-3907-v3.patch against master branch at commit dbd70f56d1926bf94bb444c918c56671c30a41e8. ATTACHMENT ID: 12871742 {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:red}-1 tests included{color}. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:red}-1 javadoc{color}. The javadoc tool appears to have generated 50 warning messages. {color:red}-1 release audit{color}. The applied patch generated 5 release audit warnings (more than the master's current 0 warnings). {color:green}+1 lineLengths{color}. The patch does not introduce lines longer than 100 {color:red}-1 core tests{color}. The patch failed these unit tests: ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.CaseStatementIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.index.MutableIndexFailureIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.OnDuplicateKeyIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.QueryDatabaseMetaDataIT Test results: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1019//testReport/ Release audit warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1019//artifact/patchprocess/patchReleaseAuditWarnings.txt Javadoc warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1019//artifact/patchprocess/patchJavadocWarnings.txt Console output: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1019//console This message is automatically generated.
    **label:** code-design
15. **body:** Attaching v4 patch with a cleaner fix for the test failure.
    **label:** code-design
16. +1
17. **body:** {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12871879/PHOENIX-3907-v4.patch against master branch at commit 65bd849becbe961bf63c1ac1d88f0cc6a4201931. ATTACHMENT ID: 12871879 {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:red}-1 tests included{color}. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:red}-1 javadoc{color}. The javadoc tool appears to have generated 50 warning messages. {color:red}-1 release audit{color}. The applied patch generated 5 release audit warnings (more than the master's current 0 warnings). {color:red}-1 lineLengths{color}. The patch introduces the following lines longer than 100: + MutationPlan plan = compiler.compile(Collections.singletonList(tableRef), emptyCF, null, null, ts); + this.upperBoundTimeStamp = table.getUpdateCacheFrequency()!=0 ? QueryConstants.UNSET_TIMESTAMP : upperBoundTimeStamp; {color:red}-1 core tests{color}. The patch failed these unit tests: ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.SaltedViewIT ./phoenix-core/target/failsafe-reports/TEST-

org.apache.phoenix.end2end.index.MutableIndexFailureIT ./phoenix-core/target/failsafe-reports/TEST-org.apache.phoenix.end2end.TimezoneOffsetFunctionIT Test results: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1030//testReport/ Release audit warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1030//artifact/patchprocess/patchReleaseAuditWarnings.txt Javadoc warnings: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1030//artifact/patchprocess/patchJavadocWarnings.txt Console output: https://builds.apache.org/job/PreCommit-PHOENIX-Build/1030//console This message is automatically generated.
**label:** code-design

18. FAILURE: Integrated in Jenkins build Phoenix-master #1648 (See [https://builds.apache.org/job/Phoenix-master/1648/]) PHOENIX-3907 Use LATEST_TIMESTAMP when UPDATE_CACHE_FREQUENCY is not (thomas: rev 7cb16d4dd7f5fe11a10dfe4a58eb2bced313b6c1) * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/CreateTableCompiler.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/schema/MetaDataClient.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/schema/TableRef.java * (edit) phoenix-core/src/main/java/org/apache/phoenix/compile/StatementContext.java