Item 362
git_comments:

1. **comment:** Temporarily disable filters on Timestamp columns due to ORC-611 new Object[] { "orc", "timestamp", "2018-06-29T10:02:34.000000", "2018-06-29T15:02:34.000000" }, new Object[] { "orc", "timestamptz", "2018-06-29T10:02:34.000000+00:00", "2018-06-29T10:02:34.000000-07:00" },
   label: code-design
2. Temporarily disable filters on Timestamp columns due to ORC-611
3. Temporarily disable filters on Timestamp columns due to ORC-611 .equals("`tsTz`", Type.TIMESTAMP, Timestamp.from(tsTzPredicate.toInstant())) .equals("`ts`", Type.TIMESTAMP, Timestamp.from(tsPredicate.toInstant()))

git_commits:

1. **summary:** ORC: Disable predicate pushdown for timestamp type (#1069)
   message: ORC: Disable predicate pushdown for timestamp type (#1069) * Correctly handle timestamps less than epoch * Disable ORC pushdown for timestamp types because of ORC-611

github_issues:

github_issues_comments:

github_pulls:

1. **title:** ORC: Disable predicate pushdown for timestamp type
   body: In https://github.com/apache/incubator-iceberg/pull/983#issuecomment-633808288 we discovered issues with ORC predicate pushdown for timestamp types, where timestamps less than epoch were not handled correctly. https://github.com/apache/incubator-iceberg/commit/ed91355d5559858cd665560f92c6d2dfdc4a6885 fixes this and updates the test case to test for both pre epoch and post epoch values After fixing this, I also found that ORC column stats for timestamp are only stored to millisecond precisions (with round down) and hence predicates on timestamp types with sub-millisecond literals can potentially be evaluated incorrectly. This is being fixed in https://issues.apache.org/jira/browse/ORC-611. So for the time being, I have disabled pushdown for timestamp types in https://github.com/apache/incubator-iceberg/commit/63bc62b0707f9fea220759b12a8a544aa63eb59e
2. **title:** ORC: Disable predicate pushdown for timestamp type
   body: In https://github.com/apache/incubator-iceberg/pull/983#issuecomment-633808288 we discovered issues with ORC predicate pushdown for timestamp types, where timestamps less than epoch were not handled correctly. https://github.com/apache/incubator-iceberg/commit/ed91355d5559858cd665560f92c6d2dfdc4a6885 fixes this and updates the test case to test for both pre epoch and post epoch values After fixing this, I also found that ORC column stats for timestamp are only stored to millisecond precisions (with round down) and hence predicates on timestamp types with sub-millisecond literals can potentially be evaluated incorrectly. This is being fixed in https://issues.apache.org/jira/browse/ORC-611. So for the time being, I have disabled pushdown for timestamp types in https://github.com/apache/incubator-iceberg/commit/63bc62b0707f9fea220759b12a8a544aa63eb59e
3. **title:** ORC: Disable predicate pushdown for timestamp type
   body: In https://github.com/apache/incubator-iceberg/pull/983#issuecomment-633808288 we discovered issues with ORC predicate pushdown for timestamp types, where timestamps less than epoch were not handled correctly. https://github.com/apache/incubator-iceberg/commit/ed91355d5559858cd665560f92c6d2dfdc4a6885 fixes this and updates the test case to test for both pre epoch and post epoch values After fixing this, I also found that ORC column stats for timestamp are only stored to millisecond precisions (with round down) and hence predicates on timestamp types with sub-millisecond literals can potentially be evaluated incorrectly. This is being fixed in https://issues.apache.org/jira/browse/ORC-611. So for the time being, I have disabled pushdown for timestamp types in https://github.com/apache/incubator-iceberg/commit/63bc62b0707f9fea220759b12a8a544aa63eb59e
4. **title:** ORC: Disable predicate pushdown for timestamp type
   body: In https://github.com/apache/incubator-iceberg/pull/983#issuecomment-633808288 we discovered issues with ORC predicate pushdown for timestamp types, where timestamps less than epoch were not handled correctly. https://github.com/apache/incubator-iceberg/commit/ed91355d5559858cd665560f92c6d2dfdc4a6885 fixes this and updates the test case to test for both pre epoch and post epoch values After fixing this, I also found that ORC column stats for timestamp are only stored to millisecond precisions (with round down) and hence predicates on timestamp types with sub-millisecond literals can potentially be evaluated incorrectly. This is being fixed in https://issues.apache.org/jira/browse/ORC-611. So for the time being, I have disabled pushdown for timestamp types in https://github.com/apache/incubator-iceberg/commit/63bc62b0707f9fea220759b12a8a544aa63eb59e
5. **title:** ORC: Disable predicate pushdown for timestamp type
   body: In https://github.com/apache/incubator-iceberg/pull/983#issuecomment-633808288 we discovered issues with ORC predicate pushdown for timestamp types, where timestamps less than epoch were not handled correctly. https://github.com/apache/incubator-iceberg/commit/ed91355d5559858cd665560f92c6d2dfdc4a6885 fixes this and updates the test case to test for both pre epoch and post epoch values After fixing this, I also found that ORC column stats for timestamp are only stored to millisecond precisions (with round down) and hence predicates on timestamp types with sub-millisecond literals can potentially be evaluated incorrectly. This is being fixed in https://issues.apache.org/jira/browse/ORC-611. So for the time being, I have disabled pushdown for timestamp types in https://github.com/apache/incubator-iceberg/commit/63bc62b0707f9fea220759b12a8a544aa63eb59e

github_pulls_comments:

1. cc @rdblue @chenjunjiedada

github_pulls_reviews:

1. Why did this change?
2. The previous logic did not correctly handle pre-epoch timestamps and resulted in off by one second error which was the primary cause of failures in #983 (`%` returns negative values if sign(a)!=sign(b) so it interfered with floorDiv which also has handling for the same). After I fixed this issue, I ran into ORC-611,

jira_issues:

1. **summary:** Incorrect min-max stats for sub-millisecond timestamps
   description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
2. **summary:** Incorrect min-max stats for sub-millisecond timestamps
   description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files

with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875

3. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
4. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
5. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
6. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
label: code-design
7. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
label: code-design
8. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
9. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
label: code-design
10. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range.

Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875

11. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
label: code-design
12. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
label: code-design
13. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
14. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
15. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
16. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875
label: code-design
17. **summary:** Incorrect min-max stats for sub-millisecond timestamps
description: The issue is related to the precision of storing timestamps: - nanoseconds for the data itself - only milliseconds for min-max statistics Both min and max are rounded to the same value, while min should be rounded down and max should be rounded up to ensure that the values are actually within that range. Repro in Hive: {code} create table tsstat (ts timestamp) stored as orc; insert into tsstat values ("1970-01-01 00:00:00.0005") select * from tsstat where ts = "1970-01-01 00:00:00.0005"; -- returned 0 rows {code} Both the Java and the C++ writer has this issue (thanks [~stigahuang] for looking them up): https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/java/core/src/java/org/apache/orc/impl/writer/TimestampTreeWriter.java#L108 https://github.com/apache/orc/blob/fea154436c37c81a16b13d879b510096cfaa2946/c%2B%2B/src/ColumnWriter.cc#L1800 I guess that there are already files with this issue in production, so I think that the only way to fix this is to hack the reader: - decrease/increase min/max stats with 1 ms after reading them from file - also be careful about the values pushed down, as the same precision loss can occur there to, eg. "WHERE ts <'1970-01-01 00:00:00.0005' AND ts > '1970-01-

01 00:00:00.0004'" shouldn't be converted to ts < "1970-01-01" AND ts > "1970-01-01" The issue was discovered during an Impala review: https://gerrit.cloudera.org/#/c/15403/1/be/src/exec/hdfs-orc-scanner.cc@875

jira_issues_comments:

1. [~csringhofer] / [~stigahuang] thanks for reporting this! Just created a PR that fixes the column stats precision issue for Timestamps (for the Java Writer initially) – also added some tests to verify the expected behaviour. [~jcamachorodriguez] [~gopalv] [~ashutoshc] thoughts? We should also add a followup ticket to change the HIVE Reader behaviour for ORC files written with existing logic.
2. I put some more thought into the patch and it seems that we can sort out the issue completely on the Reader side, avoid changing the Column stats that are written and more importantly having to support another writer version. In the latest patch, whenever we are evaluating a predicate with Timestamps, I just widen the min/max range by 1 ms to make sure the underlying ns range is included -- this solves the issue both on Java and C++ consumers using our Readers – until we feel comfortable changing the column statistics on the writer side which however will still have ms precision thus I am not sure if it does worth the extra effort.
3. I think that it would be better to fix the writer in the long run as other tools may read the stats and bump into the same issue. [~pgaref] What do you think about adding optional nanosec fields to TimestampStatistics? If it is missing then 0 could be used for MIN and 999999 for MAX.
4. [~pgaref] [~csringhofer] As I have commented in the PR, I think it is enough to work around this only on the reader side. Unless some external components are depending on the exact min/max values of Timestamp type.
5. Thanks for the comments [~wgtmac] ! I had the same feeling initially – trying to avoid another writer version. Truncating Timestamps could raise correctness issues though when used with equality, leading to wrong results returned – it seems that following a similar direction to ORC-203 is the right thing to do as a long-term solution. What other people think about this? [~gopalv] [~omalley] [~jcamachorodriguez]
6. **body:** [~pgaref] I agree that a long-term solution would be better. But I don't see why equality filtering has correctness issues. For example, if the predicate is x = 1970-01-01 00:00:00.000_5, and both min and max are 1970-01-01 00:00:00.000 Then if we fix PPD to use min = 1970-01-01 00:00:00.000_000_000 and max = 1970-01-01 00:00:00.000_999_999, the truth value for this predicate is still good.
   label: code-design
7. **body:** I think in general the problem is that we are making the assumption that the ns precision does not really make a difference in our ColumnStats while in reality there might be workloads that could benefit from higher precision. Following the Jira example lets say we insert 1970-01-01 00:00:00.000_5 {color:#172b4d}– using the Reader solution above we would have *min* 1970-01-01 00:00:00.000 and *max* 1970-01-01 00:00:00.000_999_999. Having a predicate: x = 1970-01-01 00:00:00.000 would return true-- the real min though is: 1970-01-01 00:00:00.000_5{color}
   label: code-design
8. IMO, predicate push down is only responsible for filtering out row groups that it is 100% sure will not match the predicate. A single row group contains 10K rows by default and it is highly possible that some rows are not likely matching the predicate. Therefore, returning true from predicate x = 1970-01-01 00:00:00.000 is fine because the FilterOperator in the execution engine will do its job. Though the min/max stats have trimmed nanos, the actual data does not.
9. **body:** Hey [~wgtmac], I understand why the Reader FIX would be sufficient for most of the existing cases but I am not sure why the Writer change is a bad idea. Adding nanosecond precision to the Writer could enable us do row-level filtering in the future and thus even eliminate the filter operator altogether from the op pipeline. That said, I am more than happy to split this patch in a Reader FIX and a Writer FIX if this can help back-porting efforts. Thoughts?
   label: code-design
10. [~pgaref] CMIIAW, row-level filtering operates on actual data not row-group-level column statistics only, right? I am not saying the writer change is a bad idea. A new writer version breaks all readers which are still on an older version, unless there is a strong incentive to do so. I think it is a good idea to split the patch and leave more time for the community to review the writer change.
11. **body:** > A new writer version breaks all readers which are still on an older version Why does it break them? Old versions should be still able to read the files if the new stats are optional fields. Or the readers explicitly refuse to read files from newer writers? > That said, I am more than happy to split this patch in a Reader FIX and a Writer FIX I agree with splitting it to two patches. The reader fix seems straightforward (adding 1 ms or 999999 ns to max), but for the writer I see 2 solutions with their own pros and cons: 1. Round max up to the next millisec instead of down. 2. Store the nanosecond precision in new optional fields. 1 has the advantage of fixing this issue even in old readers when reading new files, while 2 allows more precise filtering (more than 10000 rows for a single millisecond doesn't seem a likely use case but it also doesn't seem impossible).
    label: code-design
12. **body:** bq. A new writer version breaks all readers which are still on an older version After looking at the protobuf changes, I can't imagine a good enough way this will break any older readers as such (for existing data or new data). The protobuf binary compatibility should take care of all existing data and existing readers. Older readers continue to have the same bug when reading old data or new data. So, the only question is whether we want the new data to be readable by older readers without modification (i.e counter-act the bug in the non-upgraded readers, by adjusting the rounding the other way - max of 1.01 will become 2 - 0.99, rather than 1 + 0.01). bq. That said, I am more than happy to split this patch in a Reader FIX and a Writer FIX if this can help back-porting efforts. This will need backporting to 1.5.x, which patch should not have the writer change, but can retain the proto change without a writer change.
    label: code-design
13. Guys thanks for the comments, had an offline discussion with [~gopalv] and it seems that even the writer version change can be avoided as the new binary files are compatible with existing readers. In he latest patch (PR) the protobuf is updated with the extra minNanos and maxNanos without the writer change and the nano precision is handled transparently: * In the case of old ORC files without TS ns precision, minNanos is by default 0 and for maxNanos, Max ms is incremented by 1. * In the case of newly written ORC files with ns precision, minNanos is the value written and maxValue is the diff: (1ms - nanos) as stored. As a result, when the maxValue is available we just deduct the diff from the max+1 ms to get the max ns while being backwards compatible (when no maxValue ns is available max ms is still +1 ms but with no deduction – solving the reported problem for older ORC files). Thoughts ?
14. I just committed this. Thank you, Panos!
15. Released as part of 1.6.4.