Item 128
git_comments:

git_commits:

1. **summary:** [SPARK-6544][build] Increment Avro version from 1.7.6 to 1.7.7
   message: [SPARK-6544][build] Increment Avro version from 1.7.6 to 1.7.7 Fixes bug causing Kryo serialization to fail with Avro files in between stages. https://issues.apache.org/jira/browse/AVRO-1476?focusedCommentId=13999249&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-13999249 Author: Dean Chen <deanchen5@gmail.com> Closes #5193 from deanchen/SPARK-6544 and squashes the following commits: 813d4c5 [Dean Chen] [SPARK-6544] [build] Increment Avro version from 1.7.6 to 1.7.7

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-6544][build] Increment Avro version from 1.7.6 to 1.7.7
   body: Fixes bug causing Kryo serialization to fail with Avro files in between stages. https://issues.apache.org/jira/browse/AVRO-1476?focusedCommentId=13999249&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-13999249

github_pulls_comments:

1. Can one of the admins verify this patch?
2. We should probably sync this to the underlying Hadoop version. However I know this already doesn't match it and I suspect that bumping a maintenance release won't hurt. If you have a moment would be good to see which version of Hadoop uses this or else what if any incompatible changes may exist. I call it out since I had some issues with older Avro releases in the past.
3. Here are the 3 issues resolved in 1.7.7 compared to 1.7.6 https://issues.apache.org/jira/browse/AVRO/fixforversion/12326041/?selectedTab=com.atlassian.jira.jira-projects-plugin:version-summary-panel [AVRO-1632](https://issues.apache.org/jira/browse/AVRO-1632) - is marked as not a problem so there was no code change [AVRO-1439](https://issues.apache.org/jira/browse/AVRO-1439) - looks like a new API so it is new code as opposed to modification of existing code [AVRO-1476](https://issues.apache.org/jira/browse/AVRO-1476) - is the issue causing the GenericRecord problem and the motivation behind the version upgrade So it looks like a low risk change. The Hadoop MR client doesn't appear to specify the avro package version so it should be on 1.7.7 since Avro 1.7.7 was released on 07/23/2014. https://github.com/apache/hadoop/blob/3c9181722b05a9192f5440ea8f3f77231f84eac6/hadoop-mapreduce-project/hadoop-mapreduce-client/pom.xml
4. Yeah I think it's pretty safe, LGTM.
5. Yeah LGTM - I think bumping maintenance releases like this has usually been fine for us with Avro. It may be best not merge this into 1.3 though, since we typically keep dependencies fixed in patch releases.
6. ok to test
7. [Test build #29303 has started](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/29303/consoleFull) for PR 5193 at commit [`813d4c5`](https://github.com/apache/spark/commit/813d4c57c0ae3a7842bac66739df5d03b358572a). - This patch merges cleanly.
8. [Test build #29303 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/29303/consoleFull) for PR 5193 at commit [`813d4c5`](https://github.com/apache/spark/commit/813d4c57c0ae3a7842bac66739df5d03b358572a). - This patch **passes all tests**. - This patch merges cleanly. - This patch adds no public classes.
9. Test PASSed. Refer to this link for build results (access rights to CI server needed): https://amplab.cs.berkeley.edu/jenkins//job/SparkPullRequestBuilder/29303/ Test PASSed.

10. @pwendell this is a blocker for any jobs using Avro and Kryo that passes GenericData.Records between stages so I there is a good reason to merge it in to 1.3 for the next patch release. The other change from Avro 1.7.6 to 1.7.7 is just a new API so should be very low risk.
11. It's a fair point - in some cases we have made minor (patch) release updates in our own patch versions. @srowen what do you think? It can cause some friction for packagers if dependencies change in these patch releases, but this upgrade may be alright.
12. Okay after reviewing the changes to Avro I've decided to backport this. I looked a bit more and in other cases we have done fairly small dependency updates like this provided the changes in the dependency have been minimal between versions. Also thanks @deanchen for taking a look at the changes in Avro between these versions, it made it a lot easier to quickly scan through them.
13. Thanks!

github_pulls_reviews:

jira_issues:

1. **summary:** Make position field of org.apache.avro.Schema not transient.
   description: Referring to: https://github.com/apache/avro/blob/trunk/lang/java/avro/src/main/java/org/apache/avro/Schema.java#L358 [~kryzthov] did some research into possible causes/reasons for this field to be marked as transient but was unable to find any reason. The org.apache.avro.Schema class is not marked as serializable so this transient field serves no purpose. This transient field can cause odd behaviors with external serialization frameworks (and the built-in java serialization framework) when trying to serialize schemas.

jira_issues_comments:

1. The one line patch to fix this.
2. The change looks fine by me. Can someone else comment on the need for transient?
3. The question is, if we made it serializable, would we want to serialize this field? I'd suggest not. What application is harmed by this declaration? Avro reflect ignores fields that are declared as transient. I suspect the origin of this declaration is that I was looking at the schema that reflect generated for Schema. However there are now a number of other fields in Schema.java which should also be declared transient and are not. For example, RecordSchema#fieldMap, Name#full, etc. These, like Field#position, are computed from other fields and cached, but were added after Field#position and not declared transient. Since we're not consistent about transient declarations in Schema.java, removing this one should be harmless.
4. I would think you would want to serialize this field. Without serializing it, the position of each field in a record would be lost. Unless I'm misreading this, not having correct position information for a field in a record would cause all fields to be written to a single position. Since transient fields will get reset to their type's default value, to support regenerating this field post-serialization, either the marker value for "uninitialized value" should be changed to 0 (the default value for int) or the need to regenerate this cached variable should be indicated elsewhere.
5. An example: http://java67.blogspot.com/2012/08/what-is-transient-variable-in-java.html
6. Robert, you're right, with Java serialization, we probably would want to serialize this field, even though its value is derived entirely from other fields. +1 on the provided patch.
7. +1 to this as well. Kryo serialization, the most common internal serialization framework for Spark also ignores this field, making it impossible to pass GenericData.Records through Spark stages.
8. This patch looks good to me. Tests run fine. If nobody objects, I'd like to push it soon. [~cutting] is that ok?
9. +1
10. Commit 1598151 from [~cutting] in branch 'avro/trunk' [https://svn.apache.org/r1598151] AVRO-1476. Remove transient declaration from Schema.Field#position. Contributed by Robert Chu.
11. I committed this. Thanks, Robert!
12. FAILURE: Integrated in AvroJava #452 (See [https://builds.apache.org/job/AvroJava/452/]) AVRO-1476. Remove transient declaration from Schema.Field#position. Contributed by Robert Chu. (cutting: rev 1598151) * /avro/trunk/CHANGES.txt * /avro/trunk/lang/java/avro/src/main/java/org/apache/avro/Schema.java