Item 121
git_comments:

1. For double-zipped RDDs, the batches can be iterators from other PairDeserializer, instead of lists. We need to convert them to lists if needed.
2. Tests for SPARK-21985

git_commits:

1. **summary:** [SPARK-21985][PYSPARK] PairDeserializer is broken for double-zipped RDDs
message: [SPARK-21985][PYSPARK] PairDeserializer is broken for double-zipped RDDs ## What changes were proposed in this pull request? (edited) Fixes a bug introduced in #16121 In PairDeserializer convert each batch of keys and values to lists (if they do not have `__len__` already) so that we can check that they are the same size. Normally they already are lists so this should not have a performance impact, but this is needed when repeated `zip`'s are done. ## How was this patch tested? Additional unit test Author: Andrew Ray <ray.andrew@gmail.com> Closes #19226 from aray/SPARK-21985.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-16589][PYTHON] Chained cartesian produces incorrect number of records
body: ## What changes were proposed in this pull request? Fixes a bug in the python implementation of rdd cartesian product related to batching that showed up in repeated cartesian products with seemingly random results. The root cause being multiple iterators pulling from the same stream in the wrong order because of logic that ignored batching. `CartesianDeserializer` and `PairDeserializer` were changed to implement `_load_stream_without_unbatching` and borrow the one line implementation of `load_stream` from `BatchedSerializer`. The default implementation of `_load_stream_without_unbatching` was changed to give consistent results (always an iterable) so that it could be used without additional checks. `PairDeserializer` no longer extends `CartesianDeserializer` as it was not really proper. If wanted a new common super class could be added. Both `CartesianDeserializer` and `PairDeserializer` now only extend `Serializer` (which has no `dump_stream` implementation) since they are only meant for *de*serialization. ## How was this patch tested? Additional unit tests (sourced from #14248) plus one for testing a cartesian with zip.

github_pulls_comments:

1. **[Test build #69571 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/69571/consoleFull)** for PR 16121 at commit [`a0e3652`](https://github.com/apache/spark/commit/a0e36522175bed10a6309b2fe2d37793d746584b). * This patch **fails Python style tests**. * This patch merges cleanly. * This patch adds no public classes.
2. **[Test build #69573 has finished](https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/69573/consoleFull)** for PR 16121 at commit [`ad43e31`](https://github.com/apache/spark/commit/ad43e3150e557e378d9d24fdad1f509c6b7bab79). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
3. It's pretty tricky to make the chained CartesianDeserializer work, maybe it's easier to have a workaround in the RDD.cartesian() to add an _reserialize() between chained cartesian (or zipped), it will be less performant, but given cartesian() is already super slow, I will not worry about it. The current patch may still be wrong in case of chained DartesianDeserializer and PairSerializer, for example, a.cartesian(b.zip(c)) (have not verified yet)
4. @davies I was trying to make minimal changes to `PairDeserializer`, but you are right it needs changed also. I'll update the PR shortly.
5. @davies I suggested workaround before but I remember that @holdenk had some reservations. Moreover it would have to be done proactively for all (?) calls. For example [SPARK-17756](https://issues.apache.org/jira/browse/SPARK-17756) seems to hit a similar problem.

6. **[Test build #69587 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/69587/consoleFull)** for PR 16121 at commit [`6e3d9d0`] (https://github.com/apache/spark/commit/6e3d9d01b7d31fe2d874338d0690d1442fbe3995). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
7. I was hesistant with the previous PR since it seemed like we didn't fully understand why we were changing what we were at the time, I can try and take a closer look at this over the next few days if it is in a good place for that to happen.
8. **[Test build #69674 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/69674/consoleFull)** for PR 16121 at commit [`36e3876`] (https://github.com/apache/spark/commit/36e387628bfaca4922b0555787ff8c91ba6b0d93). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
9. @davies, @zero323, and @holdenk this is in a good place for review if you want to take a look.
10. **[Test build #69865 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/69865/consoleFull)** for PR 16121 at commit [`12f3ab0`] (https://github.com/apache/spark/commit/12f3ab0bdd0f19bc5ec67f5d82f1e242f870e302). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
11. LGTM, merging into master and 2.1/2.0 branch, thanks!
12. This PR seems to have introduced a bug, which I have reported here: https://issues.apache.org/jira/browse/SPARK-21985 Any thoughts, @aray? Can the check in question simply be removed, or is there a better solution to consider?
13. I'll take a look, sorry about that.

github_pulls_reviews:

1. Even though this is internal it might make sense to have a docstring for this since were changing its behaviour.
2. Maybe we should document this a bit given that we had problems with the implementation. (e.g. expand on the "Due to batching, we can't use the Java cartesian method." comment from `rdd.py` to explain how this is intended to function).
3. Maybe consider adding a comment here explaining why the interaction of batching & product

jira_issues:

jira_issues_comments: