Item 344
git_comments:

1. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
2. * * Internal class indicating a concurrent merge occurred during the new bulk import.
3. * * Check if splits were specified in plan when a concurrent merge occurred. If so, throw error * back to user since retrying won't help. If not, then retry.
4. retry if a merge occurs
5. merge happened after the mapping was generated and before the table lock was acquired

git_commits:

1. **summary:** Retry new Bulk import on merge. Fixes #471 (#1367)
   message: Retry new Bulk import on merge. Fixes #471 (#1367) * Created AccumuloBulkMergeException for handling merges with Retry maxWait up to 2 Minutes

github_issues:

1. **title:** Retry when conncurrent merge happens in bulk import.
   body: This is follow up work to #436. Currently, the new bulk import API detects concurrent merges and throws an exception. It would be better to retry.
2. **title:** Retry when conncurrent merge happens in bulk import.
   body: This is follow up work to #436. Currently, the new bulk import API detects concurrent merges and throws an exception. It would be better to retry.
3. **title:** Retry when conncurrent merge happens in bulk import.
   body: This is follow up work to #436. Currently, the new bulk import API detects concurrent merges and throws an exception. It would be better to retry.

github_issues_comments:

1. I just saw this running random walk concurrent test on 2.0. ``` java.lang.Exception: Error running node Concurrent.xml at org.apache.accumulo.testing.randomwalk.Module.visit(Module.java:370) at org.apache.accumulo.testing.randomwalk.Framework.run(Framework.java:48) at org.apache.accumulo.testing.randomwalk.Framework.main(Framework.java:92) Caused by: java.lang.Exception: Error running node ct.BulkImport at org.apache.accumulo.testing.randomwalk.Module.visit(Module.java:370) at org.apache.accumulo.testing.randomwalk.Module$1.call(Module.java:303) at org.apache.accumulo.testing.randomwalk.Module$1.call(Module.java:298) at java.util.concurrent.FutureTask.run(FutureTask.java:266) at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) at java.util.concurrent.FutureTask.run(FutureTask.java:266) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at org.apache.accumulo.fate.util.LoggingRunnable.run(LoggingRunnable.java:35) at java.lang.Thread.run(Thread.java:748) Caused by: org.apache.accumulo.core.client.AccumuloException: Concurrent merge happened at org.apache.accumulo.core.clientImpl.TableOperationsImpl.doFateOperation(TableOperationsImpl.java:385) at org.apache.accumulo.core.clientImpl.TableOperationsImpl.doFateOperation(TableOperationsImpl.java:342) at org.apache.accumulo.core.clientImpl.TableOperationsImpl.doBulkFateOperation(TableOperationsImpl.java:329) at org.apache.accumulo.core.clientImpl.bulk.BulkImport.load(BulkImport.java:142) at org.apache.accumulo.testing.randomwalk.concurrent.BulkImport.visit(BulkImport.java:130) ... 9 more Caused by: ThriftTableOperationException(tableId:a49, tableName:null, op:BULK_IMPORT, type:OTHER, description:Concurrent merge happened) at org.apache.accumulo.core.master.thrift.FateService$waitForFateOperation_result$waitForFateOperation_resultStandardScheme.read(FateService.java:4729) at org.apache.accumulo.core.master.thrift.FateService$waitForFateOperation_result$waitForFateOperation_resultStandardScheme.read(FateService.java:4698) at org.apache.accumulo.core.master.thrift.FateService$waitForFateOperation_result.read(FateService.java:4624) at org.apache.thrift.TServiceClient.receiveBase(TServiceClient.java:88) at org.apache.accumulo.core.master.thrift.FateService$Client.recv_waitForFateOperation(FateService.java:155) at org.apache.accumulo.core.master.thrift.FateService$Client.waitForFateOperation(FateService.java:140) at org.apache.accumulo.core.clientImpl.TableOperationsImpl.waitForFateOperation(TableOperationsImpl.java:292) at org.apache.accumulo.core.clientImpl.TableOperationsImpl.doFateOperation(TableOperationsImpl.java:358) ... 13 more ```

github_pulls:

1. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
2. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
3. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips

a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.

4. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
   label: documentation
5. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
6. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
7. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
8. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
   label: code-design
9. **title:** ACCUMULO-4813 New bulk import process and API
   body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.
   label: code-design
10. **title:** ACCUMULO-4813 New bulk import process and API
    body: This is a big change to how bulk import works. This new API does the following for bulk import. * Computes mapping of files to tablets on the client side. This mapping is stored in HDFS as JSON (so its human readable). Stored mapping in HDFS so it does need to be read in memory on master. * The Fate operation streams the mapping from HDFS (does not load into memory) and metadata table and sends oneway thrift load message to tablet servers. This makes the fate operation very fast. Using this new API 10K files were loaded in 20 secs on a single node with 100 tablets and the FATE op took 5 secs. The new API and functionality is complete, however the following still needs to be done. * Make metadata scan split tolerant (so it never skips a tablet) The way GC scans is split tolerant. * Decide how to make shell support this. * Possibly allow user to pass their own mapping file. * Retry when concurrent merge happens. * Document new API * Support bulk import to offline tables * When removing load flags, only scan portion of metadata table where loads happened. Currently scans entire table range in metadata table. After this PR is finished with review, will open issues for these and anything else that comes up.

github_pulls_comments:

1. Since this PR was merged, should the ACCUMULO-4813 JIRA issue be closed now?

github_pulls_reviews:

1. I think this should return ImportExecutorOptions

2. **body:** Need to update this javadoc
   label: documentation
3. Need to put the new type at the END of the list so the existing numbers don't change.
4. Need to put the new type at the END of the list so the existing numbers don't change.
5. Looks like find/replace caught some unintentional javadoc. We didn't modify this file
6. **body:** This should probably send tableId instead of name.
   label: code-design
7. **body:** Should probably move this configuration type parsing into the ConfigurationTypeHelper.
   label: code-design

jira_issues:

jira_issues_comments: