Item 38
git_comments:

1. row + OperationWithAttributes.attributes

git_commits:

1. **summary:** HBASE-8393 Testcase TestHeapSize#testMutations is wrong (Jeffrey)
   message: HBASE-8393 Testcase TestHeapSize#testMutations is wrong (Jeffrey) git-svn-id: https://svn.apache.org/repos/asf/hbase/trunk@1476022 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Testcase TestHeapSize#testMutations is wrong
   description: I happened to check this test case and there are several existing errors to make it pass. You can reproduce the test case failure by adding a new field into Mutation, the test case will either fail on a 64 bit system or 32 bit one. Below are errors I found in the test case: 1) The test case is using {code}row=new byte[]{0}{code} which is an array with length=1 while ClassSize.estimateBase can only calculate base class size(without counting field array length) 2) Add ClassSize.REFERENCE twice in the following code because ClassSize.estimateBase adds all reference fields already. {code}expected += ClassSize.align(ClassSize.TREEMAP + ClassSize.REFERENCE);{code} 3) ClassSize.estimateBase round up the sum of length of reference fields + primitive fields + Array while Mutation.MUTATION_OVERHEAD aligns the sum of length of a different set of fields. Therefore, there will be round up differences for class Increment because it introduces a new reference field TimeRange tr when the test case runs on a 32bit and 64 bit system. {code} ... long prealign_size = coeff[0] + align(coeff[1] * ARRAY) + coeff[2] * REFERENCE; // Round up to a multiple of 8 long size = align(prealign_size); ... {code}
2. **summary:** Testcase TestHeapSize#testMutations is wrong
   description: I happened to check this test case and there are several existing errors to make it pass. You can reproduce the test case failure by adding a new field into Mutation, the test case will either fail on a 64 bit system or 32 bit one. Below are errors I found in the test case: 1) The test case is using {code}row=new byte[]{0}{code} which is an array with length=1 while ClassSize.estimateBase can only calculate base class size(without counting field array length) 2) Add ClassSize.REFERENCE twice in the following code because ClassSize.estimateBase adds all reference fields already. {code}expected += ClassSize.align(ClassSize.TREEMAP + ClassSize.REFERENCE);{code} 3) ClassSize.estimateBase round up the sum of length of reference fields + primitive fields + Array while Mutation.MUTATION_OVERHEAD aligns the sum of length of a different set of fields. Therefore, there will be round up differences for class Increment because it introduces a new reference field TimeRange tr when the test case runs on a 32bit and 64 bit system. {code} ... long prealign_size = coeff[0] + align(coeff[1] * ARRAY) + coeff[2] * REFERENCE; // Round up to a multiple of 8 long size = align(prealign_size); ... {code}
3. **summary:** Testcase TestHeapSize#testMutations is wrong
   description: I happened to check this test case and there are several existing errors to make it pass. You can reproduce the test case failure by adding a new field into Mutation, the test case will either fail on a 64 bit system or 32 bit one. Below are errors I found in the test case: 1) The test case is using {code}row=new byte[]{0}{code} which is an array with length=1 while ClassSize.estimateBase can only calculate base class size(without counting field array length) 2) Add ClassSize.REFERENCE twice in the following code because ClassSize.estimateBase adds all reference fields already. {code}expected += ClassSize.align(ClassSize.TREEMAP + ClassSize.REFERENCE);{code} 3) ClassSize.estimateBase

round up the sum of length of reference fields + primitive fields + Array while Mutation.MUTATION_OVERHEAD aligns the sum of length of a different set of fields. Therefore, there will be round up differences for class Increment because it introduces a new reference field TimeRange tr when the test case runs on a 32bit and 64 bit system. {code} ... long prealign_size = coeff[0] + align(coeff[1] * ARRAY) + coeff[2] * REFERENCE; // Round up to a multiple of 8 long size = align(prealign_size); ... {code}
label: code-design

4. **summary:** Testcase TestHeapSize#testMutations is wrong
   description: I happened to check this test case and there are several existing errors to make it pass. You can reproduce the test case failure by adding a new field into Mutation, the test case will either fail on a 64 bit system or 32 bit one. Below are errors I found in the test case: 1) The test case is using {code}row=new byte[]{0}{code} which is an array with length=1 while ClassSize.estimateBase can only calculate base class size(without counting field array length) 2) Add ClassSize.REFERENCE twice in the following code because ClassSize.estimateBase adds all reference fields already. {code}expected += ClassSize.align(ClassSize.TREEMAP + ClassSize.REFERENCE);{code} 3) ClassSize.estimateBase round up the sum of length of reference fields + primitive fields + Array while Mutation.MUTATION_OVERHEAD aligns the sum of length of a different set of fields. Therefore, there will be round up differences for class Increment because it introduces a new reference field TimeRange tr when the test case runs on a 32bit and 64 bit system. {code} ... long prealign_size = coeff[0] + align(coeff[1] * ARRAY) + coeff[2] * REFERENCE; // Round up to a multiple of 8 long size = align(prealign_size); ... {code}
5. **summary:** Testcase TestHeapSize#testMutations is wrong
   description: I happened to check this test case and there are several existing errors to make it pass. You can reproduce the test case failure by adding a new field into Mutation, the test case will either fail on a 64 bit system or 32 bit one. Below are errors I found in the test case: 1) The test case is using {code}row=new byte[]{0}{code} which is an array with length=1 while ClassSize.estimateBase can only calculate base class size(without counting field array length) 2) Add ClassSize.REFERENCE twice in the following code because ClassSize.estimateBase adds all reference fields already. {code}expected += ClassSize.align(ClassSize.TREEMAP + ClassSize.REFERENCE);{code} 3) ClassSize.estimateBase round up the sum of length of reference fields + primitive fields + Array while Mutation.MUTATION_OVERHEAD aligns the sum of length of a different set of fields. Therefore, there will be round up differences for class Increment because it introduces a new reference field TimeRange tr when the test case runs on a 32bit and 64 bit system. {code} ... long prealign_size = coeff[0] + align(coeff[1] * ARRAY) + coeff[2] * REFERENCE; // Round up to a multiple of 8 long size = align(prealign_size); ... {code}
6. **summary:** Testcase TestHeapSize#testMutations is wrong
   description: I happened to check this test case and there are several existing errors to make it pass. You can reproduce the test case failure by adding a new field into Mutation, the test case will either fail on a 64 bit system or 32 bit one. Below are errors I found in the test case: 1) The test case is using {code}row=new byte[]{0}{code} which is an array with length=1 while ClassSize.estimateBase can only calculate base class size(without counting field array length) 2) Add ClassSize.REFERENCE twice in the following code because ClassSize.estimateBase adds all reference fields already. {code}expected += ClassSize.align(ClassSize.TREEMAP + ClassSize.REFERENCE);{code} 3) ClassSize.estimateBase round up the sum of length of reference fields + primitive fields + Array while Mutation.MUTATION_OVERHEAD aligns the sum of length of a different set of fields. Therefore, there will be round up differences for class Increment because it introduces a new reference field TimeRange tr when the test case runs on a 32bit and 64 bit system. {code} ... long prealign_size = coeff[0] + align(coeff[1] * ARRAY) + coeff[2] * REFERENCE; // Round up to a multiple of 8 long size = align(prealign_size); ... {code}
7. **summary:** Testcase TestHeapSize#testMutations is wrong
   description: I happened to check this test case and there are several existing errors to make it pass. You can reproduce the test case failure by adding a new field into Mutation, the test case will either fail on a 64 bit system or 32 bit one. Below are errors I found in the test case: 1) The test case is using {code}row=new byte[]{0}{code} which is an array with length=1 while ClassSize.estimateBase can only calculate base class size(without counting field array length) 2) Add ClassSize.REFERENCE twice in the following code because ClassSize.estimateBase adds all reference fields already. {code}expected += ClassSize.align(ClassSize.TREEMAP + ClassSize.REFERENCE);{code} 3) ClassSize.estimateBase round up the sum of length of reference fields + primitive fields + Array while Mutation.MUTATION_OVERHEAD aligns the sum of length of a different set of fields. Therefore, there

will be round up differences for class Increment because it introduces a new reference field TimeRange tr when the test case runs on a 32bit and 64 bit system. {code} ... long prealign_size = coeff[0] + align(coeff[1] * ARRAY) + coeff[2] * REFERENCE; // Round up to a multiple of 8 long size = align(prealign_size); ... {code}

jira_issues_comments:

1. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12579960/hbase-8393.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 3 new or modified tests. {color:green}+1 hadoop2.0{color}. The patch compiles against the hadoop 2.0 profile. {color:green}+1 javadoc{color}. The javadoc tool did not generate any warning messages. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 1.3.9) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:green}+1 lineLengths{color}. The patch does not introduce lines longer than 100 {color:green}+1 site{color}. The mvn site goal succeeds with this patch. {color:red}-1 core tests{color}. The patch failed these unit tests: org.apache.hadoop.hbase.security.access.TestAccessController org.apache.hadoop.hbase.master.TestTableLockManager Test results: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//testReport/ Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-protocol.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-client.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-examples.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-hadoop1-compat.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-prefix-tree.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-common.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-server.html Findbugs warnings: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//artifact/trunk/patchprocess/newPatchFindbugsWarningshbase-hadoop-compat.html Console output: https://builds.apache.org/job/PreCommit-HBASE-Build/5388//console This message is automatically generated.
2. {code} - cl = Increment.class; {code} Can we test heap size for Increment class ?
3. **body:** [~ted_yu] The problem for Increment class is that it has a new reference field. Because Mutation.MUTATION_OVERHEAD rounds up partial reference fields while ClassSize.estimateBase round up the sum of length of reference fields + primitive fields + Array. Therefore, there will be round up differences when the test case runs on a 32bit and 64 bit system. I can add special logic inside the test case for different OS while it's prone to fail in the future when a new field is introduced again.
   label: code-design
4. +1 on current patch then.
5. Integrated to 0.95 and trunk.
6. Integrated in hbase-0.95 #163 (See [https://builds.apache.org/job/hbase-0.95/163/]) HBASE-8393 Testcase TestHeapSize#testMutations is wrong (Jeffrey) (Revision 1476024) Result = FAILURE tedyu : Files : * /hbase/branches/0.95/hbase-client/src/main/java/org/apache/hadoop/hbase/client/Mutation.java * /hbase/branches/0.95/hbase-server/src/test/java/org/apache/hadoop/hbase/io/TestHeapSize.java
7. Integrated in hbase-0.95-on-hadoop2 #81 (See [https://builds.apache.org/job/hbase-0.95-on-hadoop2/81/]) HBASE-8393 Testcase TestHeapSize#testMutations is wrong (Jeffrey) (Revision 1476024) Result = FAILURE tedyu : Files : * /hbase/branches/0.95/hbase-client/src/main/java/org/apache/hadoop/hbase/client/Mutation.java * /hbase/branches/0.95/hbase-server/src/test/java/org/apache/hadoop/hbase/io/TestHeapSize.java
8. Integrated in HBase-TRUNK-on-Hadoop-2.0.0 #511 (See [https://builds.apache.org/job/HBase-TRUNK-on-Hadoop-2.0.0/511/]) HBASE-8393 Testcase TestHeapSize#testMutations is wrong (Jeffrey) (Revision

1476022) Result = FAILURE tedyu : Files : * /hbase/trunk/hbase-client/src/main/java/org/apache/hadoop/hbase/client/Mutation.java * /hbase/trunk/hbase-server/src/test/java/org/apache/hadoop/hbase/io/TestHeapSize.java

9. Integrated in HBase-TRUNK #4081 (See [https://builds.apache.org/job/HBase-TRUNK/4081/]) HBASE-8393 Testcase TestHeapSize#testMutations is wrong (Jeffrey) (Revision 1476022) Result = SUCCESS tedyu : Files : * /hbase/trunk/hbase-client/src/main/java/org/apache/hadoop/hbase/client/Mutation.java * /hbase/trunk/hbase-server/src/test/java/org/apache/hadoop/hbase/io/TestHeapSize.java