Item 340
git_comments:

1. * *
2. * Need to consider the following cases for inclusive and exclusive (lgcf:locality group column family set, cf:column family set) lgcf and cf are * disjoint lgcf and cf are the same cf contains lgcf lgcf contains cf lgccf and cf intersect but neither is a subset of the other
3. * 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.
4. * * @param localityGroup * @param env
5. * 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.
6. * 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.
7. **comment:** TODO time reading all data
   label: requirement
8. * 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.
9. * *
10. clear immediately so mutations can be garbage collected
11. the last map in the array is the default locality group
12. **comment:** this method is synchronized because it reuses objects to avoid allocation, currently, the method that calls this is synchronized so there is no loss in parallelism.... synchronization was added here for future proofing
    label: code-design
13. **comment:** TODO uncomment following when ACCUMULO-1628 is fixed seekLocalityGroups(iter1.deepCopy(null));
    label: code-design

git_commits:

1. **summary:** ACCUMULO-112 added locality group support to in memory map
   message: ACCUMULO-112 added locality group support to in memory map

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Investigate partitioning in memory map by locality group
   description: Currently the in memory map is not partitioned by locality group. This could negatively impact scan and minor compaction performance. Would like to run some experiments to understand the performance implications. Partitioning by locality group could negatively impact insert performance, it could go from O(log(R)+log(C)) to O(L * (log(R)+log(C))) in the worst case. L is the number of locality groups, R is the number of rows and C is the number of columns. The worst case is where each mutation has a change for each locality group. Currently the in memory map is a map of maps. Like the following. {noformat} map<row, map<col, val>> {noformat} Could conceptually change this to one of the following. The first is best for scans, that access some locality groups, and minor compactions. The second is good for inserts where the mutation covers all locality groups, because the row is only looked up once. {noformat} map<localityGroup, map<row, map<col, val>>> {noformat} {noformat} map<row, map<localityGroup, map<col, val>>> {noformat} The Accumulo native map is implemented using C++,STL, JNI, and with thread locking in java.

jira_issues_comments:

1. I ran some test with random data. The data was of the following format : {noformat} <16 digit rand hex> <4 digit hex> <4 digit rand hex> <50 byte random value> {noformat} There were 32 column families, 0000 to 001f. For the experiment 32,768 rows with 32 columns were inserted, creating 1,048,576 entries. The number of locality groups were varied and minor compaction times were recorded. Column families were evenly divided among locality groups. Below are the minor compaction times. ||Num Locality Groups||Minor Compaction Time||Relative Time|| |1 (default LG)|3.5 secs|1.0| |2|4.3 secs|1.2| |4|6.4 secs|1.8| |8|9.4 secs|2.7| |16|16.4 secs|4.7| |32|30.2 secs|8.6| Since the data was written to an unpartitioned in memory map, the insert times should have been the same. Once the in memory map is partitioned, it would be useful to track ingest time and minor compaction time. LZO was used for compression.
2. If ACCUMULO-519 is implemented then it will supercede this ticket.
3. I have a working version of this [on github|https://github.com/keith-turner/accumulo/tree/ACCUMULO-112]. It still needs some polishing and test, but its mostly done. I ran the same test I ran earlier with 32K rows each having 32 column families. I used snappy for this test. The column "Scan One CF" is the time it took to read one of the 32 column families. The column "Scan CF:CQ" is the time it took to read one column family and one column qualifier. This scan usually returned 0 to 2 entries. ||Num Locality Groups||Write Time||Scan One CF Time||Scan CF:CQ Time||Flush Time|| |1|2.21 secs|0.99 secs|0.79 secs|2.07 secs| |2|2.27 secs|0.71 secs|0.51 secs|2.19 secs| |4|2.35 secs|0.43 secs|0.20 secs|2.21 secs| |8|2.48 secs|0.33 secs|0.12 secs|2.33 secs| |16|2.86 secs|0.26 secs|0.07 secs|2.56 secs| |32|3.85 secs|0.24 secs|0.05 secs|2.85 secs| Below the data is normalized per column. Each cell is divided by the minimum in its column. ||Num Locality Groups||Write Time||Scan One CF Time||Scan CF:CQ Time||Flush Time|| |1|1.00|4.13|15.80|1.00| |2|1.03|2.96|10.20|1.06| |4|1.06|1.79|4.00|1.07| |8|1.12|1.38|2.40|1.13| |16|1.29|1.08|1.40|1.24| |32|1.74|1.00|1.00|1.38|
4. Commit 0608e32f8b09f926e40677d1e23ccedd0d6e088d in branch refs/heads/master from [~keith_turner] [https://git-wip-us.apache.org/repos/asf?p=accumulo.git;h=0608e32] ACCUMULO-112 added locality group support to in memory map