Item 175
git_comments:

1. * * java.lang.ref.ReferenceQueue utility class. * @param <K> * @param <V>
2. * * A SoftReference derivative so that we can track down what keys to remove.
3. * * A Map that uses Soft Reference values internally. Use as a simple cache. * * @param <K> key class * @param <V> value class
4. * * A SortedMap implementation that uses Soft Reference values * internally to make it play well with the GC when in a low-memory * situation. Use as a cache where you also need SortedMap functionality. * * @param <K> key class * @param <V> value class
5. Cannot connect to master or it is not running. Sleep & retry

git_commits:

1. **summary:** HBASE-953 Enable BLOCKCACHE by default [WAS -> Reevaluate HBASE-288 block caching work....?]
   message: HBASE-953 Enable BLOCKCACHE by default [WAS -> Reevaluate HBASE-288 block caching work....?] git-svn-id: https://svn.apache.org/repos/asf/hadoop/hbase/trunk@708253 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Enable BLOCKCACHE by default [WAS -> Reevaluate HBASE-288 block caching work....?]
   description: Go back and take another look at the Tom White work. We've gotten boost in writing and scanning because of J-D work. HBASE-288 looks like it boosts sequential reads and perhaps random read a little. Take another look.

jira_issues_comments:

1. Enabling cache, in PE tests I see 4X improvement in sequential read test. My guess is that scan speed will improve again too and that compactions should go the faster also. Downside is that I'm OOME'ing running random read test. Digging in some, we're creating lots of instances of the BlockFSDataStream. Each instance has its own ReferenceMap. Each ReferenceMap has 1-3 entries when blocksize is 64MB. References are cleared only when we put or get. I'm thinking this is not enough; that caches are not being cleared fast enough. Trying to dig in more.
2. PE random read also ran a good big slower.
3. Using SoftSorteMap instead of commons ReferenceMap seems to do better. Still not good enough though; I get "OOME: GC Time and OutOfMemoryError" which doesn't exactly kill it.. it hobbles along. I see random reading we're getting 90% cache hits if run after a sequential read which means cache has been warmed but the random reads should be going faster than they are. They're slowed by all the GCing I'm seeing (I have gc logging enabled). Trying w/ 1.6.0_10 JVM. Was using 1.7. If random reads were just as fast as without block reads, it would be worth enabling by default because of the faster sequential reads, scans and compactions.
4. I have been using 1.6.0_04 with concurrent sweep GC. Under modest random read workload, w/ block caching, it's faster.
5. Rong-en: Thanks for the tip. I'll try some loading tests using it. It might be better with soft references. With 1.6.0_10, with 8 concurrent clients, random reads completed. Trying again with single client.. but also thinking on it on way home, smaller blocks make more sense. I cut it down to 16k as suggested by jgray

6. So, you are talking about using smaller block size in block cache or hdfs block size?
7. Rong-en: I'm looking to enable BLOCKCACHE in HColumnDescriptor as the default. Currently its off. The size I'm playing with is this one: hbase.hstore.blockCache.blockSize I ran with 1.6.0_10 JDK and 16k blocksize and all single-client runs ran as well as an overnight 16 concurrent client test. All passed. Running some more tests to see if I can make an OOME happen.
8. 1.6.0_10 works fine. java version "1.7.0-ea" Java(TM) SE Runtime Environment (build 1.7.0-ea-b31) Java HotSpot(TM) Server VM (build 14.0-b01, mixed mode) Will OOME. Stuck doing full GCs. Works if I enable the JVM flag which says don't OOME if too many full GCs.
9. Commit comment below {code} HBASE-953 Enable BLOCKCACHE by default [WAS -> Reevaluate HBASE-288 block caching work....?] Change that makes BLOCKCACHE on by default but caching blocks of 16k rather than 64k. Means that upgrading, users must update their hbase-default.xml to pick up the smaller value else OOME issues. This patch may introduce OOME issues anyways. Need to watch out for it (Was find on 1.6.0_10 but on an early 1.7.0 was easy to OOME). This patch removes commons-collections.jar. Was only added for its ReferenceMap which is replaced in this patch by a version of what used to be called SoftSortedMap (renamed as SoftValueSortedMap); the ReferenceMap replacement is SoftValueMap. Use our own because I want to be able to watch evictions if issues with OOMEs and RM is all private when it runs the ReferenceQueue clean up (RM also had features we don't need). M conf/hbase-default.xml Update hbase.hstore.blockCache.blockSize; make 16k instaed of 64k. D lib/commons-collections-3.2.jar Removed. Not used. A src/test/org/apache/hadoop/hbase/util/SoftValueSortedMapTest.java Rename of SoftSortedMapTest. D src/test/org/apache/hadoop/hbase/util/SoftSortedMapTest.java Removed M src/java/org/apache/hadoop/hbase/HColumnDescriptor.java Enable blockcaching as default. M src/java/org/apache/hadoop/hbase/io/BlockFSInputStream.java Use SoftValueMap instead of ReferenceMap. A src/java/org/apache/hadoop/hbase/util/ReferenceQueueUtil.java ReferenceQueue utility class. A src/java/org/apache/hadoop/hbase/util/SoftValueMap.java Like SoftValueSortedMap but not sorted (based on it). A src/java/org/apache/hadoop/hbase/util/SoftValue.java Added. Was internal class of SoftSortedMap. A src/java/org/apache/hadoop/hbase/util/SoftValueSortedMap.java Rename of SoftSortedMap. A src/java/org/apache/hadoop/hbase/util/SoftSortedMap.java Like SoftValueSortedMap but a plain Map. M src/java/org/apache/hadoop/hbase/client/HConnectionManager.java SoftSortedMap was renamed SoftValueSortedMap. {code}
10. Passes all tests locally. I'll keep running loading tests over the weekend but I'd like to commit this. Any chance of a review?
11. Review: there are some long lines in HCM but +1 if corrected.
12. Tried with jdk1.7.0b38 (as opposed to the b31 used above) and it held up. Lots of full GC's but weathered the PE tests w/o OOME. J-D gave my patch a review and said fine except for some line-lengths which I've fixed.
13. Committed.