Item 170
git_comments:

git_commits:

1. **summary:** HBASE-62 Allow user add arbitrary key/value pairs to table and column descriptors; Remove an exception catch I'd left in place, remove unused imports
   message: HBASE-62 Allow user add arbitrary key/value pairs to table and column descriptors; Remove an exception catch I'd left in place, remove unused imports git-svn-id: https://svn.apache.org/repos/asf/hadoop/hbase/trunk@677603 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
2. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
3. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
   label: code-design
4. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
5. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
6. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
   label: code-design
7. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
8. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
9. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
   description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
10. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
    label: code-design
11. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.

label: code-design

12. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
13. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
14. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
15. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
    label: code-design
16. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
17. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
    label: code-design
18. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
19. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
20. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
    label: code-design
21. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
22. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
    label: code-design
23. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
24. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
25. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.
26. **summary:** [hbase] Allow user add arbitrary key/value pairs to table and column descriptors
    description: Folks have asked if they can tag columns and tables with markings of their own designation. Examples include 'type' and 'descriptiion'.

jira_issues_comments:

1. A possibility is to use a language neutral format such as JSON. A collection of key/value pairs would then simply be called an "object". Furthermore, this should integrate well with jaql, a query language we recently announced for JSON data and posted further information at http://www.jaql.org. In fact, we've implemented several functions in the language to manage HBase data. As a result, type information, through the use of JSON, is serialized into HBase byte array column values. We did not consider descriptions for tables, but I suppose this is reasonable to store in the catalog.

2. Changing priority to minor. The need for this feature is not as urgent as it once was, and there are other possible mechanisms for achieving the same goal.
3. **body:** Changing priority to trivial because demand for this request has been significantly less than in the past. I.e., trying to use priority (whose particular values do not fit this specific need) to arrange issues so that their priority reflects the urgency for a particular feature or bug fix.
   label: code-design
4. Implemented by modifying HTableDescriptor and HColumnDescriptor to incorporate a MapWritable that uses ImmutableBytesWritables for keys and values. A number of issues related to greater descriptor flexibility will be linked shortly, and I'll issue a patch that addresses them all.
5. Sounds good Andrew. You want to get your changes into 0.2 Andrew?
6. **body:** I'd like to see these changes go in for 0.2, if possible. Changes are contained largely to HTableDescriptor and HColumnDescriptor. The most signification change elsewhere is support in HRegion for read only tables. Patch attached. On second thought using a MapWritable was kind of ugly, so I just used a map with byte[] keys and values and the utilities for reading/writing byte arrays in Byte instead. Basically the same thing but the code is cleaner. The patch passes all 'ant test' tests for me. These items are outstanding: * Unit test for read-only tables -- New test TestReadOnly * Unit test for user table and column metadata -- Probably can modify TestHTable for this * Migration script to deal with HTableDescriptor changes I'll work on the unit tests.
   label: code-design
7. Patch looks great Andrew. Just a few comments. + Byte arrays as HashMap keys do object identity which is probably not what you want. Use ImmutableBytesWritable as key or a TreeMap and provide the Bytes.comparable if you want the keys to equate when their content is the same. + I like the way you do the getValue override whereby if you use a String key to ask for a value, you get it back as a String. + The statement in getValue javadoc about encoding may not be true in all cases -- e.g. user may not have set value as UTF-8?: ' * @return The value as a UTF-8 encoded byte array.' Hey, so what migration needs to be done since HTD and HCD seem to be self-migrating? You want to rewrite the .META. tables before startup? Can I help out w/ migration?
8. Thanks for the patch review. I have attached another patch which addresses the comments. * Use TreeMaps with Bytes.BYTES_COMPARATOR. Thanks for the advice. * Fixed outdated Javadoc from a previous attempt. Regarding HTableDescriptor and HColumnDescriptor migration, HCD has been self-migrating for a while but HTD is not yet. The hbase-62 patch includes changes which make HTD self-versioning, but HTD.readFields is not compatible with what is on disk today. So HTDs for all tables need to be rewritten as part of migration.
9. Patch looks good. One nitpick would be that rather than Map, you'd take SortedMap when being passed Map of key/value pairs in HRI to underline fact that it needs to be a TreeMap type-thingy.. but thats not a biggy. Can do it on commit. Do you think the migration would just be the rewriting of the HTD component of regioninfo in .META.? I suppose you want to get the migration in as part of this patch? I can do a sketch of how it'd run if you like. Just say. Good stuff Andrew.
10. **body:** I attached another patch that uses SortedMap instead of Map. For migration the proposed HTD.readFields needs an integer (written by .writeInt()) ahead of the rest of the serialized HTD to process the old format. I have tried to make migration easy. All that needs to be done is a simple prepend of a serialized integer to the serialized HTDs in .META., but I don't know how migration scripts work so maybe that is not simple or a complete rewrite is no problem...
    label: code-design
11. **body:** Thanks for new patch Andrew. I've been playing around with the migration. Its that old story where you need the old class around to deserialize but the new class is in the way. I've started up a little package under o.a.h.h.util named migration.v4. In it, I'll add classes need to do the v3 to v4 migration. WIll need to iterate over all .META. regions using new tools I need to add to MetaUtils. Will get the info:regioninfo column bytes and feed them to a subclass of HRI, one that has a diffferent deserializer. This new deserializer inserts a subclass of HTD that also has a different deserializer, one that can read v3 HTDs. It copies the v3 to a v4 HTD. Then we call the default HRI serializer -- which will invoke the v4 HTD serializer -- and insert back into .META. One hiccup is that we might have to do the v4 migration ahead of the v3 migration in tests. I'm a little worried that subclassing HRI and HTD in migration is too brittle. Will think on it over the w/e.
    label: code-design
12. Thanks for the help with migration issues Stack. I've attached another patch and cleaned up some of the old ones. This latest one I think is fully baked. I took your suggestion to use ImmutableBytesWritable, because it does seem like the more Hadoop-y thing to do. I've also added additional test cases to

TestTable (read only tables) and TestHTable (HTD and HCD attributes). All of the tests in the 'ant test' suite pass for me including the modified ones except for Migration of course.

13. Pulling this into 0.2. At a minimum, JK needs the HTD to be verisioned for HBASE-696
14. Updated patch so will apply to trunk. Still need to do the migration work this issue needs.
15. **body:** Updated patch for latest trunk. Also I made a package o.a.h.h.util.migration.v0_2_0 and put the old HTableDescriptor, HColumnDescriptor, and BloomFilter classes from 0.1 there. Because migration from 0.1.X to 0.2.0 will be explicitly done via these classes, I removed (now unnecessary) backward compatability code from the trunk versions of HTD.readFields and HCD.readFields.
    label: code-design
16. Is this patch essential for 0.2.0 or can it be moved to 0.2.1 and 0.3.0 ? I'd like to freeze 0.2.0 as soon as possible so we can make a release candidate.
17. **body:** I've been working on the necessary migration. Its ugly mixing old and new versions of about ten classes. Almost done though. We could punt bringing in this patch but the migration would still be ugly, just postponed. Might be uglier even if lots of on-filesystems in 0.3 hbase. This (painful) patch opens our easily doing other HTD extensions because it introduces HTD versioning (e.g. moving table schema out of HTD). Also adds read-only hbase table. Its kinda sweet. Regards migrations, after bringing in this patch, I think we should start over with migrations; they're complicated enough now involving 5 distinct steps. It would mean that you could not go from old hbase 0.16.x through to 0.3. You'd have to install 0.2 to bring you up to 0.2 hbase; then install 0.3 hbase to migrate from 0.2 to 0.3.
    label: code-design
18. With recent commits, the latest patch will no longer apply. I'll work on it so that it will.
19. Attached new patch that applies cleanly to trunk. I left out any migration bits because stack is doing his own thing.
20. **body:** Committed a simplification of migration against this issue. Previously migration was a 5 step process. Ensuring each individual step had the right set of classes to hand proved too much for my small brain. Complexity was compounded by fact that some classes were self-migrating. Things got extra messy when test migration did a forward migrate, junked the version file, and then ran a new forward migration top the old. To get this patch in, I first stripped migration to a minimum. Now, the migration script in TRUNK will only migrate between an hbase 0.1.x install and an hbase 0.2.x install. This removed one step. That leaves the addition of the regionhistorian column family and the bloom filter cleanup. Plan is to make the migration one-step doing bloomfilter cleanup and migration of HTD, etc., all in the one step. Going forward, we shouldn't be too bad now the important classes all have versions but if we think we're going to run into this awkward mix of classes in the future, we should do something like have each step run in its own JVM invocation so we have chance to doctor CLASSPATH per migration step.
    label: code-design
21. This patch adds a few edits to Andrew's last patch and then adds migration.
22. **body:** Here's a few more notes on the patch. The migration is ugly. Adds new util.migration.v5 package into which I put pre-v5 versions of classes. Migrating I use these classes to read in the serialized info and then copy fields out of the old versions of the classes into new instances of their replacements. I added versioning to HRegionInfo while I was at it in case we need to change its format someday. {code} M src/test/org/apache/hadoop/hbase/TestTable.java (testReadOnlyTable): Added. M src/test/org/apache/hadoop/hbase/client/TestHTable.java Test adding a user attribute. M src/java/org/apache/hadoop/hbase/HColumnDescriptor.java Up HCD version to 6. Attributes now kept in a map. (getValue, setValue): Added. Also added a bunch of setters for the base attributes: maxVersions, TTL, etc. isBloomFilterEnabled renamed as isBloomfilter. M src/java/org/apache/hadoop/hbase/BloomFilterDescriptor.java Added constructor. M src/java/org/apache/hadoop/hbase/thrift/ThriftUtilities.java isBloomFilterEnabled renamed as isBloomfilter. M src/java/org/apache/hadoop/hbase/regionserver/HStore.java isBloomFilterEnabled renamed as isBloomfilter. M src/java/org/apache/hadoop/hbase/regionserver/HAbstractScanner.java Made class protected so could be used from migration subpackage. M src/java/org/apache/hadoop/hbase/regionserver/HRegion.java Check if table is read-only. M src/java/org/apache/hadoop/hbase/rest/TableHandler.java isBloomFilterEnabled renamed as isBloomfilter. M src/java/org/apache/hadoop/hbase/HTableDescriptor.java Version HTD. New constructor that clones a passed HTD. (getValue, setValue, setReadOnly): Added. Other accessors and setters for old base attributes. M src/java/org/apache/hadoop/hbase/HRegionInfo.java Version HRI. (setTableDesc, getVersion): Added. M src/java/org/apache/hadoop/hbase/ipc/HMasterInterface.java M src/java/org/apache/hadoop/hbase/master/HMaster.java (modifyTableMeta): Added. A src/java/org/apache/hadoop/hbase/util/migration/v5/HLogEdit.java A

src/java/org/apache/hadoop/hbase/util/migration/v5/HLogKey.java A src/java/org/apache/hadoop/hbase/util/migration/v5/Memcache.java A src/java/org/apache/hadoop/hbase/util/migration/v5/RegionHistorian.java A src/java/org/apache/hadoop/hbase/util/migration/v5/HColumnDescriptor.java A src/java/org/apache/hadoop/hbase/util/migration/v5/HStore.java A src/java/org/apache/hadoop/hbase/util/migration/v5/HStoreFile.java A src/java/org/apache/hadoop/hbase/util/migration/v5/StoreFileScanner.java A src/java/org/apache/hadoop/hbase/util/migration/v5/HRegionInfo.java A src/java/org/apache/hadoop/hbase/util/migration/v5/MetaUtils.java A src/java/org/apache/hadoop/hbase/util/migration/v5/package.html A src/java/org/apache/hadoop/hbase/util/migration/v5/HStoreScanner.java A src/java/org/apache/hadoop/hbase/util/migration/v5/LogRollListener.java A src/java/org/apache/hadoop/hbase/util/migration/v5/FlushRequester.java A src/java/org/apache/hadoop/hbase/util/migration/v5/HConstants.java A src/java/org/apache/hadoop/hbase/util/migration/v5/HLog.java A src/java/org/apache/hadoop/hbase/util/migration/v5/HRegion.java Above are all pre-v5 versions of the above classes. Most of the classes are unchanged except for edits to the imports so we imported from this migration.v5 package rather than from outside this package. MetaUtils is the one exception. I changed it to use getWritables when instantiating HRegionInfos so I could make it instantiate HRIs from this package rather than use o.a.h.h.HRI. M src/java/org/apache/hadoop/hbase/util/Migrate.java (migrateToV5, rewriteMetaHRegionInfo, updateHRegionInfo): Added. (migrateRegionDir, processRegionSubDirs, scanRootRegion): Removed. M src/java/org/apache/hadoop/hbase/client/HTable.java (getTableDescriptor): Make the returned HTD unmodifiable. M src/java/org/apache/hadoop/hbase/client/MetaScanner.java make MetaScannerVisitor protected so can get to it from migration code. A src/java/org/apache/hadoop/hbase/client/UnmodifyableHColumnDescriptor.java M src/java/org/apache/hadoop/hbase/client/HBaseAdmin.java (modifyTableMeta): Added. M src/java/org/apache/hadoop/hbase/client/UnmodifyableHTableDescriptor.java Added unsupportedoperationexception to setter overrides. {code}
label: code-design

23. I reviewed Andrew's code. Looks good to me. Any chance of a quick review of the migration code? All changes under o.a.h.h.util? Passes all tests on my box.
24. Resolving. Its a bit cheeky checking in this fat patch w/o getting the migration reviewed but then, who wants to look at migration code? I reran the test suite locally and it passed so in it goes. Thanks for the patch Andrew. Good stuff.