Item 5
git_comments:

1. * * Perform the write of a mutation given a WritePerformer. * Gather the list of write endpoints, apply locally and/or forward the mutation to * said write endpoint (deletaged to the actual WritePerformer) and wait for the * responses based on consistency level. * * @param mutations the mutations to be applied * @param consistency_level the consistency level for the write operation * @param performer the WritePerformer in charge of appliying the mutation * given the list of write endpoints (either standardWritePerformer for * standard writes or counterWritePerformer for counter writes).
2. We need to separate row mutation for standard cf and counter cf (that will be encapsulated in a CounterMutation) because it doesn't follow the same code path

git_commits:

1. **summary:** merge CASSANDRA-2457 from 0.8
   message: merge CASSANDRA-2457 from 0.8 git-svn-id: https://svn.apache.org/repos/asf/cassandra/trunk@1095335 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Batch_mutate is broken for counters
   description: CASSANDRA-2384 allowed for batch_mutate to take counter and non counter operation, but the code was not updated correctly to handle that case. As it is, the code will use the first mutation in the batch list to decide whether to apply the write code path of counter or not, and will thus break if those are mixed.

jira_issues_comments:

1. Attaching patch against 0.8. Basically this makes SP.mutate() handle a list of mixed RowMutation and CounterMutation (but then each mutation use the right path). This is needed since a batch_mutate() can now mix those. This has an unfortunate consequence however in that I don't think there is a way to distinguish the writeStats of standard insert of the ones of counter insert anymore, and thus this patch remove the latter ones (and count everything in writeStat).
2. **body:** * writeLocallyAndReplicate doesn't always perform a local mutation, so it should probably be renamed * Since mutateCounter and writeLocallyAndReplicate are symmetrical and are called depending on whether an IMutation is an instance of CounterMutation, could we move them onto IMutation, and polymorphically decide the behavior? > ...and thus this patch remove the latter ones (and count everything in writeStat). I'm fine with this, since a counter is as "real" a write as any other. _But_ I do think we should record the latencies for the replicate-on-write stage like we do for the read and mutation stages on a per column family basis. I can tackle it in a separate ticket if you'd like. PS: Thanks for removing this line!
   {code}mutations.iterator().next().getColumnFamilies().iterator().next().metadata().getDefaultValidator().isCommutative()
   {code} Thanks Sylvain!
   label: code-design
3. **body:** Attached rebased version (post-CASSANDRA-2454 in particular) bq. writeLocallyAndReplicate doesn't always perform a local mutation, so it should probably be renamed Renamed it to performWrite (since it mostly simply imply a so-called writePerformer). bq. Since mutateCounter and writeLocallyAndReplicate are symmetrical and are called depending on whether an IMutation is an instance of CounterMutation, could we move them onto IMutation, and polymorphically decide the behavior? Hum, they are not really so symmetrical. In particular writeLocallyAndReplicate (or performWrite as it is called nowadays) really is polymorphic over the IMutation used. So the only seem we we could do (at least easily) is moving some of mutateCounter in CounterMutation, but not sure it will look so great. Also, it is probably nice to keep all the code related to the write/read protocol in StorageProxy (doing otherwise would be like moving the query code out of CFStore, nobody wants that :D) bq. I'm fine with this, since a counter is as "real" a write as any other. But I do think we should record the latencies for the replicate-on-write stage like we do for the read and mutation stages on a per column family basis. I can tackle it in a separate ticket if you'd like. Make sense, but I'm also in favor of moving this to some other ticket.
   label: code-design
4. +1
5. Committed, thanks

6. Integrated in Cassandra #859 (See [https://hudson.apache.org/hudson/job/Cassandra/859/]) merge CASSANDRA-2457 from 0.8
7. Thanks Sylvain. Opened CASSANDRA-2522