Item 3
git_comments:

git_commits:

1. **summary:** move log ops to callables on a threadpoolexecutor instead of synchronizing. this prepares the way to merge multiple add() calls into a single sync.
   message: move log ops to callables on a threadpoolexecutor instead of synchronizing. this prepares the way to merge multiple add() calls into a single sync. patch by jbellis; reviewed by Jun Rao for CASSANDRA-182 git-svn-id: https://svn.apache.org/repos/asf/incubator/cassandra/trunk@798370 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
2. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
   label: code-design
3. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
   label: code-design
4. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
   label: code-design
5. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
6. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
7. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
8. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
9. **summary:** CommitLog.add doesn't really force to disk
   description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
10. **summary:** CommitLog.add doesn't really force to disk
    description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
11. **summary:** CommitLog.add doesn't really force to disk

description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
label: code-design

12. **summary:** CommitLog.add doesn't really force to disk
description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
13. **summary:** CommitLog.add doesn't really force to disk
description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
label: code-design
14. **summary:** CommitLog.add doesn't really force to disk
description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
15. **summary:** CommitLog.add doesn't really force to disk
description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
16. **summary:** CommitLog.add doesn't really force to disk
description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
17. **summary:** CommitLog.add doesn't really force to disk
description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.
18. **summary:** CommitLog.add doesn't really force to disk
description: CommitLog.add does't really force writes to disk. This could result in acked writes being lost.

jira_issues_comments:

1. 1. Added a class in SequenceFile called SyncWriter 2. New configuration option CommitLogForceLogs if you want "safe" writes. Not turning this on leaves everything else the same as before. All this does is use SyncWriter for the logger. For trunk and 0.4 we should revisit the options we want to allow on the logs, especially once we re-architect the logger to use batched forces.
2. **body:** The more I look at the commitlog code the more I'm inclined to go with my initial reaction, that we shouldn't mess with this for 0.3... :-| The FastSync option looks like A&P's attempt to deal with this. It does call force() on close, for instance. (Is that really all we need for durability?) Having two options for syncing, one of which is commented to have issues and the other is virtually entirely untested, doesn't really seem like an improvement to me. If we did go with "let's add a third option" then IMO we should be doing the force in CommitLog not the writer. updateHeader for instance does a looping write w/ a single writer; i assume only one sync for the entire method is needed, not one sync per write. For 0.3 I suggest emailing A&P off-list and see if they can clarify why fastsync has problems purging log files. If it is something simple to fix great. Otherwise let's note to "use fastsync for durable writes, but this has the drawback of not purging log files. fixing this is a priority for 0.4"
label: code-design
3. **body:** FastSync (inappropriately named, IMHO) option tells the CommitLog to use a FastConcurrentWriter. This certainly doesn't force to disk, and therefore is not an option for durable writes. If we don't add a third option for 0.3 which forces to disk, we simply have *no* durable writes. Even on blocking writes, it is possible that the data has not hit disk in any of the replica's logs. We have two options: 1. Release as is, and say "we don't offer durable writes yet" 2. Add an option to force the logs and say "this is currently a low-performing option and will be improved for 0.4" We can do the forces in CommitLog (probably a cleaner approach in terms of eliminating unnecessary forces, but updateHeader writes only for CFs encountered for the first time). The approach in CASSANDRA-182.patch seemed minimally invasive to me :-) Building a well-tested high-performance logger will take significantly more time. (Just testing the logger for correctness is non-trivial.)
label: code-design
4. **body:** FCW does force on close, leading me to believe that for whatever reason A&P felt that was all that was needed to call the option "Sync." Of course I could be reading too much into this. Always a possibility in "code archaeology." :)
label: code-design

5. Assuming we don't hear back from Avinash & Prashant, I suggest that we include a "known issues" section in the README for 0.3 noting this problem. I'm not comfortable with basing the CL on brand-new code. For 0.4 let's take a closer look at FastSync and either build on that or rip it out and go with the approach in your patch (either in CL or the writer). I'd like to see this coupled with a CL executor that force()s less than once per mutation if there is a constant stream of writes, similar to postgresql's commit_delay setting (http://www.postgresql.org/docs/8.3/static/runtime-config-wal.html).
6. I haven't heard back from A&P. I'm fine with releasing as is and saying logging doesn't work correctly yet and that it'll be fixed soon.
7. Won't fix in 0.3
8. Are you working on this, Sandeep? If not I will take it.
9. Nope, I'm not looking at this right now.
10. confirmed that naive fsync-per-write kills performance -- i'm seeing 1/10 the throughput. will try the executor approach I mentioned above.
11. **body:** I did figure how to avoid using force(true) to flush the metadata in addition to the data (2 disk forces instead of one). If you preallocate the pages for the log file, you need to force(true) only when you have to grab more pages for the log. This would of course require some extra work in the logger code so the end of the log can be tracked without depending on file size/end of file.
    label: code-design
12. the executor approach is looking really good, but that's worth keeping in mind down the road.
13. **body:** 07 switch to arrayblockingqueue; it's a little cleaner and performance seems unaffected 06 config options. increasing write threads to 32 (was 4) allows performance to be mostly decent again, especially if you crank the delay up to 10ms 05 custom CommitLogExecutorService that can fsync per multiple CL additions 04 threadpoolexecutor, just using an out-of-the-box executor as a first step 03 naive fsync-after-each-log-entry. performance _sucks._ 02 mv AbstractWriter to its own top-level class and remove redundant IFileWriter 01 handle incomplete CL entries on recover (a fairly important bug fix)
    label: code-design
14. removed old patches, attached rebased ones
15. The patch looks good to me. Thanks, Jonathan.
16. committed
17. Integrated in Cassandra #151 (See [http://hudson.zones.apache.org/hudson/job/Cassandra/151/]) Use arrayblockingqueue in commitlog executor; this cleans up the code a bit (performance is unaffected since the writes and syncs are far more expensive than any queue ops) patch by jbellis; reviewed by Jun Rao for add config options for commitlog syncing patch by jbellis; reviewed by Jun Rao for custom CommitLogExecutorService that can fsync per multiple CL additions patch by jbellis; reviewed by Jun Rao for move log ops to callables on a threadpoolexecutor instead of synchronizing. this prepares the way to merge multiple add() calls into a single sync. patch by jbellis; reviewed by Jun Rao for naive fsync-after-each-log-entry patch by jbellis; reviewed by Jun Rao for mv AbstractWriter to its own top-level class and remove redundant IFileWriter patch by jbellis; reviewed by Jun Rao for handle incomplete CL entries on recover patch by jbellis; reviewed by Jun Rao for