Item 41
git_comments:

git_commits:

1. **summary:** HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM to change WAL to extend Closeable
   message: HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM to change WAL to extend Closeable

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
2. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data

and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

3. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
4. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
5. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

6. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
7. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
8. **summary:** Concurrency issue in WAL unflushed seqId tracking
   description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
9. **summary:** Concurrency issue in WAL unflushed seqId tracking

description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
label: test

10. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
11. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
12. **summary:** Concurrency issue in WAL unflushed seqId tracking

description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

13. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
14. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
15. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened

multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

16. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
17. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
18. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not

working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

19. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
20. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
21. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO

[regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

22. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
23. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
24. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN

[regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

25. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
26. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.
27. **summary:** Concurrency issue in WAL unflushed seqId tracking
    description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS

is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

28. **summary:** Concurrency issue in WAL unflushed seqId tracking
description: I'm inspecting an interesting case where in a production cluster, some regionservers ends up accumulating hundreds of WAL files, even with force flushes going on due to max logs. This happened multiple times on the cluster, but not on other clusters. The cluster has periodic memstore flusher disabled, however, this still does not explain why the force flush of regions due to max limit is not working. I think the periodic memstore flusher just masks the underlying problem, which is why we do not see this in other clusters. The problem starts like this: {code} 2016-09-21 17:49:18,272 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=33, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-21 17:49:18,273 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} then, it continues until the RS is restarted: {code} 2016-09-23 17:43:49,356 INFO [regionserver//10.2.0.55:16020.logRoller] wal.FSHLog: Too many wals: logs=721, maxlogs=32; forcing flush of 1 regions(s): d4cf39dc40ea79f5da4d0cf66d03cb1f 2016-09-23 17:43:49,357 WARN [regionserver//10.2.0.55:16020.logRoller] regionserver.LogRoller: Failed to schedule flush of d4cf39dc40ea79f5da4d0cf66d03cb1f, region=null, requester=null {code} The problem is that region {{d4cf39dc40ea79f5da4d0cf66d03cb1f}} is already split some time ago, and was able to flush its data and split without any problems. However, the FSHLog still thinks that there is some unflushed data for this region.

jira_issues_comments:

1. In branch-1 and earlier, FSHLog.findRegionsToForceFlush() is the method which finds that region to be force-flushed. This means that the region is somehow still in the oldestUnflushedStoreSequenceIds, although the region is closed already and flushed. I suspect the issue is a race between FSHLog.startCacheFlush() and FSHLog.append() racing for the updates to the oldestUnflushedStoreSequenceIdsOfRegion map. FSHlog.append() is called from the ring buffer handler thread, and does this: {code} private void updateOldestUnflushedSequenceIds(byte[] encodedRegionName, Set<byte[]> familyNameSet, Long lRegionSequenceId) { ConcurrentMap<byte[], Long> oldestUnflushedStoreSequenceIdsOfRegion = getOrCreateOldestUnflushedStoreSequenceIdsOfRegion(encodedRegionName); for (byte[] familyName : familyNameSet) { oldestUnflushedStoreSequenceIdsOfRegion.putIfAbsent(familyName, lRegionSequenceId); } } {code} And FSHLog.startCacheFlush() does this: {code} synchronized (regionSequenceIdLock) { ConcurrentMap<byte[], Long> oldestUnflushedStoreSequenceIdsOfRegion = oldestUnflushedStoreSequenceIds.get(encodedRegionName); if (oldestUnflushedStoreSequenceIdsOfRegion != null) { for (byte[] familyName: flushedFamilyNames) { Long seqId = oldestUnflushedStoreSequenceIdsOfRegion.remove(familyName); if (seqId != null) { oldStoreSeqNum.put(familyName, seqId); } } if (!oldStoreSeqNum.isEmpty()) { Map<byte[], Long> oldValue = this.lowestFlushingStoreSequenceIds.put(encodedRegionName, oldStoreSeqNum); assert oldValue == null: "Flushing map not cleaned up for " + Bytes.toString(encodedRegionName); } if (oldestUnflushedStoreSequenceIdsOfRegion.isEmpty()) { // Remove it otherwise it will be in oldestUnflushedStoreSequenceIds for ever // even if the region is already moved to other server. // Do not worry about data racing, we held write lock of region when calling // startCacheFlush, so no one can add value to the map we removed. oldestUnflushedStoreSequenceIds.remove(encodedRegionName); } else { oldestUnflushedSequenceId = Collections.min(oldestUnflushedStoreSequenceIdsOfRegion.values()); } } {code} The actual problem is that the sub-map for the region which keeps the per-store unflushed edits gets removes from: {code} Long seqId = oldestUnflushedStoreSequenceIdsOfRegion.remove(familyName); {code} but the actual region entry is removed some time later in: {code} if (oldestUnflushedStoreSequenceIdsOfRegion.isEmpty()) { oldestUnflushedStoreSequenceIds.remove(encodedRegionName); {code} However, this comment is not correct: {code} // Do not worry about data racing, we held write lock of region when calling //

startCacheFlush, so no one can add value to the map we removed. {code} and it is likely that an append() request is still executing and will re-add the removed family to the map. Although startCacheFlush() happens under the HRegion.updatesLock().writeLock() and doMiniBatchMutation() acquires the read lock, we may give away the read lock before the ring buffer handler actually does the FSHLog.append(). We release the region lock in step 6, but only wait for the seqId assignment in step 8.

2. I had a similar problem on a customer production cluster recently. The WALs for one of the region servers (server 11) kept on accumulating, and these LOG info repeatedly showed up. {code} 2016-09-03 14:37:15,989 INFO org.apache.hadoop.hbase.regionserver.wal.FSHLog: Too many hlogs: logs=817, maxlogs=32; forcing flush of 2 regions(s): 1b86c057f80721d4fde43a303f63ebde, 32d36d4864259dc9d984326bf27dcc5e 2016-09-03 14:37:15,990 WARN org.apache.hadoop.hbase.regionserver.LogRoller: Failed to schedule flush of 1b86c057f80721d4fde43a303f63ebde, region=null, requester=null 2016-09-03 14:37:15,990 WARN org.apache.hadoop.hbase.regionserver.LogRoller: Failed to schedule flush of 32d36d4864259dc9d984326bf27dcc5e, region=null, requester=null {code} It turned out that the two regions were opened and hosted on other region servers, not on this region server. After manually moving the complaining regions from other region servers to server 11, then server 11 was able to finish the "flush". The wal files for server 11 came down right after that. I didn't had a chance to look into what the root cause was. Some of the region servers had crashed before that.
3. bq. It turned out that the two regions were opened and hosted on other region servers, not on this region server. Do you know that whether these regions were opened before on this host, then closed. If it is, it maybe the same issue. At the time of the log roller and log file cleanup, the region to force flush is already closed some time ago, but WAL still thinks that there is unflushed entries from this region.
4. In one occurrence of this issue, the region was actually non-existent. It was a region that was split into two regions, and the daughters were opened fine. But the flush continued to be blocked.
5. bq. Do you know that whether these regions were opened before on this host. I don't have access to the complete logs to confirm that now. My impression is they were.
6. This should solve the problem for branch-1. On master, we may not have the same problem since in master, we keep holding the region read lock until the existing transaction is already sync'ed. The patch makes it so that after getting the region write lock, we also wait for the ongoing transactions before starting the FSHLog.startCacheFlush() call. We already wait for all transactions to finish under the region write lock, so semantically the behavior should not change (would be good if the reviewers verify this independently).
7. Good one. Rationale is reasonable. I like how 'no change', just reorder. Trying to come up w/ a test would be hard. You know it intimately now though. Possible to manufacture at all?
8. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 15m 31s {color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s {color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s {color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 2m 23s {color} | {color:green} branch-1 passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 36s {color} | {color:green} branch-1 passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 38s {color} | {color:green} branch-1 passed with JDK v1.7.0_111 {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 1s {color} | {color:green} branch-1 passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 18s {color} | {color:green} branch-1 passed {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 2m 8s {color} | {color:red} hbase-server in branch-1 has 1 extant Findbugs warnings. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 32s {color} | {color:green} branch-1 passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 39s {color} | {color:green} branch-1 passed with JDK v1.7.0_111 {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 51s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 39s {color} | {color:green} the patch passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 39s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 39s {color} | {color:green} the patch passed with JDK v1.7.0_111 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 39s {color} | {color:green} the patch passed {color} | |

{color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 0s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 17s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s {color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 18m 27s {color} | {color:green} The patch does not cause any errors with Hadoop 2.4.0 2.4.1 2.5.0 2.5.1 2.5.2 2.6.1 2.6.2 2.6.3 2.7.1. {color} | | {color:green}+1{color} | {color:green} hbaseprotoc {color} | {color:green} 0m 17s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 26s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 33s {color} | {color:green} the patch passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 40s {color} | {color:green} the patch passed with JDK v1.7.0_111 {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 84m 42s {color} | {color:green} hbase-server in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 17s {color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 135m 6s {color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=1.11.2 Server=1.11.2 Image:yetus/hbase:date2016-09-28 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12830795/hbase-16721_v1.branch-1.patch | | JIRA Issue | HBASE-16721 | | Optional Tests | asflicense javac javadoc unit findbugs hadoopcheck hbaseanti checkstyle compile | | uname | Linux f215a66d2783 3.13.0-95-generic #142-Ubuntu SMP Fri Aug 12 17:00:09 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/patchprocess/precommit/personality/hbase.sh | | git revision | branch-1 / 96a8e8d | | Default Java | 1.7.0_111 | | Multi-JDK versions | /usr/lib/jvm/java-8-oracle:1.8.0_101 /usr/lib/jvm/java-7-openjdk-amd64:1.7.0_111 | | findbugs | v3.0.0 | | findbugs | https://builds.apache.org/job/PreCommit-HBASE-Build/3764/artifact/patchprocess/branch-findbugs-hbase-server-warnings.html | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/3764/testReport/ | | modules | C: hbase-server U: hbase-server | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/3764/console | | Powered by | Apache Yetus 0.3.0 http://yetus.apache.org | This message was automatically generated.

9. **body:** I was able to come up with a unit test for this. It is a bit involved, but basically simulates the situation by holding the threads at appropriate times. As any good unit test should, it fails on without the patch: {code} java.lang.AssertionError: Found seqId for the region which is already flushed Expected :-1 Actual :3 {code} Let me see whether we need the patch in master.
   label: test
10. Attaching the master patch as well. On master, the test does not fail, since as explained about we do not unlock the updatesLock until the transaction is fully complete (WAL.append() went through). Hence the master patch is only the test case.
11. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 19s {color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s {color} | {color:green} The patch does not contain any @author tags. {color} | | {color:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s {color} | {color:green} The patch appears to include 1 new or modified test files. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 2m 8s {color} | {color:green} branch-1 passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 34s {color} | {color:green} branch-1 passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 37s {color} | {color:green} branch-1 passed with JDK v1.7.0_111 {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 57s {color} | {color:green} branch-1 passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 18s {color} | {color:green} branch-1 passed {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 1m 56s {color} | {color:red} hbase-server in branch-1 has 1 extant Findbugs warnings. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 31s {color} | {color:green} branch-1 passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 34s {color} | {color:green} branch-1 passed with JDK v1.7.0_111 {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 47s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 31s {color} | {color:green} the patch passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 31s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 35s

{color} | {color:green} the patch passed with JDK v1.7.0_111 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 35s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 54s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 17s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s {color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 17m 21s {color} | {color:green} The patch does not cause any errors with Hadoop 2.4.0 2.4.1 2.5.0 2.5.1 2.5.2 2.6.1 2.6.2 2.6.3 2.7.1. {color} | | {color:green}+1{color} | {color:green} hbaseprotoc {color} | {color:green} 0m 16s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 12s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 29s {color} | {color:green} the patch passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 36s {color} | {color:green} the patch passed with JDK v1.7.0_111 {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 83m 21s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 17s {color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 116m 2s {color} | {color:black} {color} | \\ \\ || Reason || Tests || | Failed junit tests | hadoop.hbase.mapreduce.TestMultiTableSnapshotInputFormat | \\ \\ || Subsystem || Report/Notes || | Docker | Client=1.11.2 Server=1.11.2 Image:yetus/hbase:date2016-09-29 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12830833/hbase-16721_v2.branch-1.patch | | JIRA Issue | HBASE-16721 | | Optional Tests | asflicense javac javadoc unit findbugs hadoopcheck hbaseanti checkstyle compile | | uname | Linux abcba1203533 3.13.0-36-lowlatency #63-Ubuntu SMP PREEMPT Wed Sep 3 21:56:12 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/patchprocess/precommit/personality/hbase.sh | | git revision | branch-1 / 96a8e8d | | Default Java | 1.7.0_111 | | Multi-JDK versions | /usr/lib/jvm/java-8-oracle:1.8.0_101 /usr/lib/jvm/java-7-openjdk-amd64:1.7.0_111 | | findbugs | v3.0.0 | | findbugs | https://builds.apache.org/job/PreCommit-HBASE-Build/3765/artifact/patchprocess/branch-findbugs-hbase-server-warnings.html | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/3765/artifact/patchprocess/patch-unit-hbase-server.txt | | unit test logs | https://builds.apache.org/job/PreCommit-HBASE-Build/3765/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/3765/testReport/ | | modules | C: hbase-server U: hbase-server | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/3765/console | | Powered by | Apache Yetus 0.3.0 http://yetus.apache.org | This message was automatically generated.

12. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 21s {color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s {color} | {color:green} The patch does not contain any @author tags. {color} | | {color:green}+1{color} | {color:green} test4tests {color} | {color:green} 0m 0s {color} | {color:green} The patch appears to include 2 new or modified test files. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 3m 10s {color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 36s {color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 45s {color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 13s {color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 1m 45s {color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 28s {color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 46s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 34s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 34s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 43s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 14s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s {color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 25m 11s {color} | {color:green} Patch does not cause any errors with Hadoop 2.4.0 2.4.1 2.5.0 2.5.1 2.5.2 2.6.1 2.6.2 2.6.3 2.7.1. {color} | | {color:green}+1{color} | {color:green} hbaseprotoc {color} | {color:green} 0m 12s

{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 1s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 29s {color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 90m 11s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 17s {color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 128m 23s {color} | {color:black} {color} | \\ \\ || Reason || Tests || | Timed out junit tests | org.apache.hadoop.hbase.client.TestReplicasClient | | | org.apache.hadoop.hbase.client.TestFromClientSide | | | org.apache.hadoop.hbase.client.TestTableSnapshotScanner | | | org.apache.hadoop.hbase.client.TestMobCloneSnapshotFromClient | | | org.apache.hadoop.hbase.client.TestMobSnapshotCloneIndependence | \\ \\ || Subsystem || Report/Notes || | Docker | Client=1.11.2 Server=1.11.2 Image:yetus/hbase:7bda515 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12830834/hbase-16721_v2.master.patch | | JIRA Issue | HBASE-16721 | | Optional Tests | asflicense javac javadoc unit findbugs hadoopcheck hbaseanti checkstyle compile | | uname | Linux da73c8acacde 3.13.0-92-generic #139-Ubuntu SMP Tue Jun 28 20:42:26 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/dev-support/hbase-personality.sh | | git revision | master / 09a31bd | | Default Java | 1.8.0_101 | | findbugs | v3.0.0 | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/3766/artifact/patchprocess/patch-unit-hbase-server.txt | | unit test logs | https://builds.apache.org/job/PreCommit-HBASE-Build/3766/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/3766/testReport/ | | modules | C: hbase-server U: hbase-server | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/3766/console | | Powered by | Apache Yetus 0.3.0 http://yetus.apache.org | This message was automatically generated.

13. I'd like to see this land in time for inclusion in 1.2.4. How is it looking for ~sunday?
14. bq. I'd like to see this land in time for inclusion in 1.2.4. How is it looking for ~sunday? The patches are ready. Need a +1 for commit actually. UT failures are not related.
15. Patch is excellent. +1.
16. Pushed this. Thanks everyone.
17. FAILURE: Integrated in Jenkins build HBase-1.3-JDK7 #25 (See [https://builds.apache.org/job/HBase-1.3-JDK7/25/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev f77f1530d4cebd1679bc1c27782bc283638dbd5f) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java
18. FAILURE: Integrated in Jenkins build HBase-Trunk_matrix #1697 (See [https://builds.apache.org/job/HBase-Trunk_matrix/1697/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev bf3c928b7499797735f71974992b68c9d876b97c) * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/AbstractTestFSWAL.java * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
19. FAILURE: Integrated in Jenkins build HBase-1.1-JDK7 #1788 (See [https://builds.apache.org/job/HBase-1.1-JDK7/1788/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev 06c3dec2da32dcb588f0eb31e5db87796668bd39) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java
20. FAILURE: Integrated in Jenkins build HBase-1.2-JDK8 #31 (See [https://builds.apache.org/job/HBase-1.2-JDK8/31/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev cf374af102f139a6176d05b97201bfa8d9f687be) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java
21. FAILURE: Integrated in Jenkins build HBase-1.4 #438 (See [https://builds.apache.org/job/HBase-1.4/438/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev bf5a7aba5c0c83874f52cbd775dd280cb4a1cd49) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java * (edit) hbase-

server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java

22. FAILURE: Integrated in Jenkins build HBase-1.2-JDK7 #34 (See [https://builds.apache.org/job/HBase-1.2-JDK7/34/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev cf374af102f139a6176d05b97201bfa8d9f687be) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java
23. FAILURE: Integrated in Jenkins build HBase-1.3-JDK8 #26 (See [https://builds.apache.org/job/HBase-1.3-JDK8/26/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev f77f1530d4cebd1679bc1c27782bc283638dbd5f) * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
24. FAILURE: Integrated in Jenkins build HBase-1.1-JDK8 #1872 (See [https://builds.apache.org/job/HBase-1.1-JDK8/1872/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking (enis: rev 06c3dec2da32dcb588f0eb31e5db87796668bd39) * (edit) hbase-server/src/test/java/org/apache/hadoop/hbase/regionserver/wal/TestFSHLog.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java
25. Why AutoCloseable instead of Closeable? Thanks.
26. bq. Why AutoCloseable instead of Closeable? Yeah, it should be Closeable instead. Let me push an addendum.
27. Pushed the addendum. Thanks Duo.
28. FAILURE: Integrated in Jenkins build HBase-1.1-JDK7 #1790 (See [https://builds.apache.org/job/HBase-1.1-JDK7/1790/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev e8019307c58da488ad26555d104990cd1c005dab) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
29. FAILURE: Integrated in Jenkins build HBase-1.2-JDK8 #33 (See [https://builds.apache.org/job/HBase-1.2-JDK8/33/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev 77e25d32b3ad8863625c9d25e3ecd7526608acf6) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
30. FAILURE: Integrated in Jenkins build HBase-1.4 #440 (See [https://builds.apache.org/job/HBase-1.4/440/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev 06cc123849aefb67570f0c016829b53ab958721b) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
31. FAILURE: Integrated in Jenkins build HBase-1.2-JDK7 #36 (See [https://builds.apache.org/job/HBase-1.2-JDK7/36/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev 77e25d32b3ad8863625c9d25e3ecd7526608acf6) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
32. FAILURE: Integrated in Jenkins build HBase-1.3-JDK8 #28 (See [https://builds.apache.org/job/HBase-1.3-JDK8/28/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev 3ddff3b665dbf83f138f8ab19b4e79f391ff71a8) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
33. FAILURE: Integrated in Jenkins build HBase-1.3-JDK7 #27 (See [https://builds.apache.org/job/HBase-1.3-JDK7/27/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev 3ddff3b665dbf83f138f8ab19b4e79f391ff71a8) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
34. SUCCESS: Integrated in Jenkins build HBase-Trunk_matrix #1710 (See [https://builds.apache.org/job/HBase-Trunk_matrix/1710/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev b8ad9b17bb6365026036687a47b3586958596d68) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
35. FAILURE: Integrated in Jenkins build HBase-1.1-JDK8 #1874 (See [https://builds.apache.org/job/HBase-1.1-JDK8/1874/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev e8019307c58da488ad26555d104990cd1c005dab) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/wal/WAL.java
36. could we get a release note on this?
37. Reopening, since we have to do a quick addendum for 1.1. See HBASE-16820.

38. This addendum is needed for 1.1 without a proper fix for HBASE-16820. It is basically undoing the HRegion changes, but adding a {{waitForPreviousTransactionsComplete()}} call at the beginning instead. We end up doing two mvcc transactions, but at the end of the flush, we are guaranteed to have the mvcc read point to be advanced to the flush sequence id.
39. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 19s {color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s {color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s {color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:red}-1{color} | {color:red} mvninstall {color} | {color:red} 2m 39s {color} | {color:red} root in branch-1.1 failed. {color} | | {color:red}-1{color} | {color:red} compile {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in branch-1.1 failed with JDK v1.8.0_101. {color} | | {color:red}-1{color} | {color:red} compile {color} | {color:red} 3m 45s {color} | {color:red} hbase-server in branch-1.1 failed with JDK v1.7.0_80. {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 9s {color} | {color:green} branch-1.1 passed {color} | | {color:red}-1{color} | {color:red} mvneclipse {color} | {color:red} 0m 14s {color} | {color:red} hbase-server in branch-1.1 failed. {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in branch-1.1 failed. {color} | | {color:red}-1{color} | {color:red} javadoc {color} | {color:red} 0m 12s {color} | {color:red} hbase-server in branch-1.1 failed with JDK v1.8.0_101. {color} | | {color:red}-1{color} | {color:red} javadoc {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in branch-1.1 failed with JDK v1.7.0_80. {color} | | {color:red}-1{color} | {color:red} mvninstall {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:red}-1{color} | {color:red} compile {color} | {color:red} 0m 5s {color} | {color:red} hbase-server in the patch failed with JDK v1.8.0_101. {color} | | {color:red}-1{color} | {color:red} javac {color} | {color:red} 0m 5s {color} | {color:red} hbase-server in the patch failed with JDK v1.8.0_101. {color} | | {color:red}-1{color} | {color:red} compile {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in the patch failed with JDK v1.7.0_80. {color} | | {color:red}-1{color} | {color:red} javac {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in the patch failed with JDK v1.7.0_80. {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 7s {color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} mvneclipse {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s {color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 11m 55s {color} | {color:green} The patch does not cause any errors with Hadoop 2.4.0 2.4.1 2.5.0 2.5.1 2.5.2 2.6.1 2.6.2 2.6.3 2.7.1. {color} | | {color:red}-1{color} | {color:red} hbaseprotoc {color} | {color:red} 0m 7s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 0m 8s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:red}-1{color} | {color:red} javadoc {color} | {color:red} 0m 6s {color} | {color:red} hbase-server in the patch failed with JDK v1.8.0_101. {color} | | {color:red}-1{color} | {color:red} javadoc {color} | {color:red} 0m 7s {color} | {color:red} hbase-server in the patch failed with JDK v1.7.0_80. {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 0m 8s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 40s {color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 21m 28s {color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=1.11.2 Server=1.11.2 Image:yetus/hbase:35e2245 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12833008/hbase-16721_addendum2.branch-1.1.patch | | JIRA Issue | HBASE-16721 | | Optional Tests | asflicense javac javadoc unit findbugs hadoopcheck hbaseanti checkstyle compile | | uname | Linux 8f73a74263de 3.13.0-92-generic #139-Ubuntu SMP Tue Jun 28 20:42:26 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/patchprocess/precommit/personality/hbase.sh | | git revision | branch-1.1 / edac952 | | Default Java | 1.7.0_80 | | Multi-JDK versions | /usr/lib/jvm/java-8-oracle:1.8.0_101 /usr/lib/jvm/java-7-oracle:1.7.0_80 | | mvninstall | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/branch-mvninstall-root.txt | | compile | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/branch-compile-hbase-server-jdk1.8.0_101.txt | | compile | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/branch-compile-hbase-server-jdk1.7.0_80.txt | | mvneclipse | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/branch-mvneclipse-

hbase-server.txt | | findbugs | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/branch-findbugs-hbase-server.txt | | javadoc | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/branch-javadoc-hbase-server-jdk1.8.0_101.txt | | javadoc | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/branch-javadoc-hbase-server-jdk1.7.0_80.txt | | mvninstall | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-mvninstall-hbase-server.txt | | compile | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-compile-hbase-server-jdk1.8.0_101.txt | | javac | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-compile-hbase-server-jdk1.8.0_101.txt | | compile | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-compile-hbase-server-jdk1.7.0_80.txt | | javac | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-compile-hbase-server-jdk1.7.0_80.txt | | mvneclipse | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-mvneclipse-hbase-server.txt | | hbaseprotoc | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-hbaseprotoc-hbase-server.txt | | findbugs | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-findbugs-hbase-server.txt | | javadoc | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-javadoc-hbase-server-jdk1.8.0_101.txt | | javadoc | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-javadoc-hbase-server-jdk1.7.0_80.txt | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/testReport/ | | modules | C: hbase-server U: hbase-server | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/3971/console | | Powered by | Apache Yetus 0.3.0 http://yetus.apache.org | This message was automatically generated.

40. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 17s {color} | {color:blue} Docker mode activated. {color} | | {color:green}+1{color} | {color:green} @author {color} | {color:green} 0m 0s {color} | {color:green} The patch does not contain any @author tags. {color} | | {color:red}-1{color} | {color:red} test4tests {color} | {color:red} 0m 0s {color} | {color:red} The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 2m 52s {color} | {color:green} branch-1.1 passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 29s {color} | {color:green} branch-1.1 passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 33s {color} | {color:green} branch-1.1 passed with JDK v1.7.0_80 {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 27s {color} | {color:green} branch-1.1 passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 15s {color} | {color:green} branch-1.1 passed {color} | | {color:red}-1{color} | {color:red} findbugs {color} | {color:red} 1m 46s {color} | {color:red} hbase-server in branch-1.1 has 80 extant Findbugs warnings. {color} | | {color:red}-1{color} | {color:red} javadoc {color} | {color:red} 0m 26s {color} | {color:red} hbase-server in branch-1.1 failed with JDK v1.8.0_101. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 30s {color} | {color:green} branch-1.1 passed with JDK v1.7.0_80 {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 0m 40s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 29s {color} | {color:green} the patch passed with JDK v1.8.0_101 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 29s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 33s {color} | {color:green} the patch passed with JDK v1.7.0_80 {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 33s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 20s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} mvneclipse {color} | {color:green} 0m 16s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} whitespace {color} | {color:green} 0m 0s {color} | {color:green} The patch has no whitespace issues. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 11m 50s {color} | {color:green} The patch does not cause any errors with Hadoop 2.4.0 2.4.1 2.5.0 2.5.1 2.5.2 2.6.1 2.6.2 2.6.3 2.7.1. {color} | | {color:green}+1{color} | {color:green} hbaseprotoc {color} | {color:green} 0m 14s {color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} findbugs {color} | {color:green} 2m 0s {color} |

{color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} javadoc {color} | {color:red} 0m 23s {color} | {color:red} hbase-server in the patch failed with JDK v1.8.0_101. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 30s {color} | {color:green} the patch passed with JDK v1.7.0_80 {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red} 96m 14s {color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 24s {color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 121m 58s {color} | {color:black} {color} | \\ \\ || Reason || Tests || | Failed junit tests | hadoop.hbase.master.procedure.TestMasterFailoverWithProcedures | | Timed out junit tests | org.apache.hadoop.hbase.coprocessor.TestRegionObserverInterface | | | org.apache.hadoop.hbase.snapshot.TestSecureExportSnapshot | | | org.apache.hadoop.hbase.snapshot.TestExportSnapshot | | | org.apache.hadoop.hbase.TestIOFencing | \\ \\ || Subsystem || Report/Notes || | Docker | Client=1.11.2 Server=1.11.2 Image:yetus/hbase:35e2245 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12833446/hbase-16721_addendum2.branch-1.1.patch | | JIRA Issue | HBASE-16721 | | Optional Tests | asflicense javac javadoc unit findbugs hadoopcheck hbaseanti checkstyle compile | | uname | Linux b0fee3270185 3.13.0-92-generic #139-Ubuntu SMP Tue Jun 28 20:42:26 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux | | Build tool | maven | | Personality | /testptch/patchprocess/precommit/personality/hbase.sh | | git revision | branch-1.1 / ba6e7dd | | Default Java | 1.7.0_80 | | Multi-JDK versions | /usr/lib/jvm/java-8-oracle:1.8.0_101 /usr/lib/jvm/java-7-oracle:1.7.0_80 | | findbugs | v3.0.0 | | findbugs | https://builds.apache.org/job/PreCommit-HBASE-Build/4028/artifact/patchprocess/branch-findbugs-hbase-server-warnings.html | | javadoc | https://builds.apache.org/job/PreCommit-HBASE-Build/4028/artifact/patchprocess/branch-javadoc-hbase-server-jdk1.8.0_101.txt | | javadoc | https://builds.apache.org/job/PreCommit-HBASE-Build/4028/artifact/patchprocess/patch-javadoc-hbase-server-jdk1.8.0_101.txt | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/4028/artifact/patchprocess/patch-unit-hbase-server.txt | | unit test logs | https://builds.apache.org/job/PreCommit-HBASE-Build/4028/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/4028/testReport/ | | modules | C: hbase-server U: hbase-server | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/4028/console | | Powered by | Apache Yetus 0.3.0 http://yetus.apache.org | This message was automatically generated.

41. Pushed addendum2 to 1.1. Other branches do not need it.
42. FAILURE: Integrated in Jenkins build HBase-1.1-JDK8 #1881 (See [https://builds.apache.org/job/HBase-1.1-JDK8/1881/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev 382f88ae87490cf0860aa63e104bf7e8273827c4) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java
43. FAILURE: Integrated in Jenkins build HBase-1.1-JDK7 #1797 (See [https://builds.apache.org/job/HBase-1.1-JDK7/1797/]) HBASE-16721 Concurrency issue in WAL unflushed seqId tracking - ADDENDUM (enis: rev 382f88ae87490cf0860aa63e104bf7e8273827c4) * (edit) hbase-server/src/main/java/org/apache/hadoop/hbase/regionserver/HRegion.java
44. Bulk closing issues after 1.1.8 release.