Item 32
git_comments:

1. * * Sets the maximum time before the buffer is automatically flushed. * @param timeoutMs The maximum number of milliseconds how long records may be buffered * before they are flushed. Set to 0 to disable. * @param timerTickMs The number of milliseconds between each check if the * timeout has been exceeded. Must be 100ms (as defined in * {@link #MIN_WRITE_BUFFER_PERIODIC_FLUSH_TIMERTICK_MS}) * or larger to avoid performance problems.
2. * * Having the timer tick run more often that once every 100ms is needless and will * probably cause too many timer events firing having a negative impact on performance.
3. * * Disable periodic flushing of the write buffer.
4. * * Returns the current periodic flush timeout value in milliseconds. * @return The maximum number of milliseconds how long records may be buffered before they * are flushed. The value 0 means this is disabled.
5. * * Returns the current periodic flush timertick interval in milliseconds. * @return The number of milliseconds between each check if the timeout has been exceeded. * This value only has a real meaning if the timeout has been set to > 0
6. * * Sets the maximum time before the buffer is automatically flushed checking once per second. * @param timeoutMs The maximum number of milliseconds how long records may be buffered * before they are flushed. Set to 0 to disable.
7. If we have the need for a timer and there is none we start it
8. Nothing to flush
9. The first record in the writebuffer has been in there too long --> flush
10. Set via the setter because it does value validation and starts/stops the TimerTask
11. No need to flush yet
12. If something changed we stop the old Timer.
13. Create Timer running as Daemon.
14. Both parameters have minimal values.
15. Stop any running Periodic Flush timer.
16. * * Set the max timeout before the buffer is automatically flushed.
17. * * Set the TimerTick how often the buffer timeout if checked.
18. 0 == Disabled 1 second
19. Reenable periodic flushing, a flush seems to take about 1 second so we wait for 2 seconds and it should have finished the flush.
20. The BufferedMutatorImpl corrects illegal values (indirect via BufferedMutatorParams)
21. The BufferedMutatorImpl corrects illegal values (direct via setter)
22. The BufferedMutatorParams does nothing with the value
23. Flush ASAP Check every 100ms Write buffer set to much larger than the single record
24. The timerTick should fire every 100ms, so after twice that we must have seen at least 1 tick and we should see an automatic flush
25. ----- Insert, NO flush, MUST flush automatically
26. ----- Insert, flush immediately, MUST NOT flush automatically
27. Ensure it does not flush twice
28. Wait for at least 1 timerTick, we should see NO flushes.
29. ----- DISABLE AUTO FLUSH, Insert, NO flush, MUST NOT flush automatically
30. Verify if BufferedMutator has the right settings.

git_commits:

1. **summary:** HBASE-19486 Periodically ensure records are not buffered too long by BufferedMutator **message:** HBASE-19486 Periodically ensure records are not buffered too long by BufferedMutator Signed-off-by: Chia-Ping Tsai <chia7712@gmail.com>

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
2. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
3. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
4. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
   label: documentation
5. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine.

The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

6. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
   label: code-design
7. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
8. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
9. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
   description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
   label: documentation
10. **summary:** Periodically ensure records are not buffered too long by BufferedMutator

description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

11. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
12. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
13. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
    label: code-design
14. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed

implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

15. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
16. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
    label: documentation
17. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
18. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
19. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful

of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

20. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
    label: code-design
21. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
22. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
23. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
    label: code-design
24. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into

HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

25. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
26. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
27. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
28. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
29. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine.

The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

30. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
31. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
32. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
33. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
    label: code-design
34. **summary:** Periodically ensure records are not buffered too long by BufferedMutator

description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

35. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
36. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
37. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
38. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

39. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
40. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
41. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
42. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
43. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed

implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

44. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
label: code-design
45. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
46. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
47. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
48. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful

of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
label: code-design

49. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
label: code-design
50. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
51. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
52. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
53. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into

HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

54. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
55. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
56. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
57. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
58. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine.

The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

59. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
60. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
61. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
62. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
63. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we

ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

64. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
65. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
66. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
67. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
68. **summary:** Periodically ensure records are not buffered too long by BufferedMutator

description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

69. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
70. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
71. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
72. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

73. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
74. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
75. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
76. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
77. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
    description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed

implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.

78. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
79. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
80. **summary:** Periodically ensure records are not buffered too long by BufferedMutator
description: I'm working on several projects where we are doing stream / event type processing instead of batch type processing. We mostly use Apache Flink and Apache Beam for these projects. When we ingest a continuous stream of events and feed that into HBase via a BufferedMutator this all works fine. The buffer fills up at a predictable rate and we can make sure it flushes several times per second into HBase by tuning the buffer size. We also have situations where the event rate is unpredictable. Some times because the source is in reality a batch job that puts records into Kafka, sometimes because it is the "predictable in production" application in our testing environment (where only the dev triggers a handful of events). For these kinds of use cases we need a way to 'force' the BufferedMutator to automatically flush any records in the buffer even if the buffer is not full. I'll put up a pull request with a proposed implementation for review against the master (i.e. 3.0.0). When approved I would like to backport this to the 1.x and 2.x versions of the client in the same (as close as possible) way.
label: code-design

jira_issues_comments:

1. I put up a patch via github https://github.com/apache/hbase/pull/69
2. Niels: HBase doesn't accept pull request. Please attach patch here.
3. [~yuzhihong@gmail.com] I have attached the changes as a diff/patch file.
4. **body:** {code} + throw new UnsupportedOperationException("The BufferedMutator::setWriteBufferMaxLinger has not been implemented"); {code} The double colon seems to be from C++. Please mention MIN_WRITE_BUFFER_MAX_LINGER in the javadoc. {code} + default long getWriteBufferMaxLinger() { {code} It would be clearer if the time unit is part of the method name.
label: documentation
5. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 12s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 45s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 20s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 28s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 4m 57s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 20s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 49s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 20s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 20s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 29s{color} | {color:red} hbase-client: The patch generated 8 new + 49 unchanged - 1 fixed = 57 total (was 50) {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} shadedjars {color} | {color:green} 4m 35s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 59m 43s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.1 2.6.2 2.6.3 2.6.4 2.6.5 2.7.1 2.7.2 2.7.3 2.7.4 or 3.0.0-alpha4. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 56s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 10s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 79m 34s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12901758/HBASE-19486-20171212-2117.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 7dd11530e557 3.13.0-129-generic #178-Ubuntu SMP Fri Aug 11 12:48:20 UTC 2017 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 / 11467ef111 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/10391/artifact/patchprocess/diff-checkstyle-hbase-client.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10391/testReport/ | | modules | C: hbase-client U: hbase-client | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10391/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

6. **body:** Fixed the checkstyle issues in the file I modified.
   label: code-design
7. Run yetus again.
8. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 9s{color} | {color:blue} Docker mode activated. {color} | | {color:red}-1{color} | {color:red} patch {color} | {color:red} 0m 8s{color} | {color:red} HBASE-19486 does not apply to master. Rebase required? Wrong Branch? See https://yetus.apache.org/documentation/0.6.0/precommit-patchnames for help. {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12902628/HBASE-19486-20171218-1229.patch | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10525/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.
9. **body:** @[~yuzhihong@gmail.com]: # The :: notation indeed looks like the C++ way of writing. I intended to make it look like the "new" [Java 8 Method References|https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html] # I renamed the mentioned getter to getWriteBufferMaxLingerMs to indicate you will be getting milliseconds.
   label: documentation
10. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 11s{color} | {color:blue} Docker

mode activated. {color} | | {color:red}-1{color} | {color:red} patch {color} | {color:red} 0m 3s{color} | {color:red} HBASE-19486 does not apply to master. Rebase required? Wrong Branch? See https://yetus.apache.org/documentation/0.6.0/precommit-patchnames for help. {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12902635/HBASE-19486-20171218-1300.patch | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10528/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

11. Rebased on current master
12. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 9s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 49s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 20s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 29s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 11s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 19s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 53s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 29s{color} | {color:red} hbase-client: The patch generated 2 new + 24 unchanged - 6 fixed = 26 total (was 30) {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} shadedjars {color} | {color:green} 4m 44s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 19m 36s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 18s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 44s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 8s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 39m 32s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12902796/HBASE-19486-20171219-0933.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 6f0de242be35 3.13.0-129-generic #178-Ubuntu SMP Fri Aug 11 12:48:20 UTC 2017 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / 7a7e55b601 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/10552/artifact/patchprocess/diff-checkstyle-hbase-client.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10552/testReport/ | | modules | C: hbase-client U: hbase-client | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10552/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.
13. **body:** Checkstyle fixes
    label: code-design

14. | (/) *{color:green}+1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 10s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 30s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 27s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 34s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 53s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 23s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 23s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 30s{color} | {color:green} hbase-client: The patch generated 0 new + 24 unchanged - 6 fixed = 24 total (was 30) {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} shadedjars {color} | {color:green} 4m 55s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 19m 11s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 18s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 42s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 8s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 41m 15s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12902808/HBASE-19486-20171219-1026.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 85b2e715491d 3.13.0-133-generic #182-Ubuntu SMP Tue Sep 19 15:49:21 UTC 2017 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / 03e79b7994 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10553/testReport/ | | modules | C: hbase-client U: hbase-client | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10553/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.
15. Mind introducing trivial change to hbase-server module to trigger QA run ?
16. **body:** Ok. Is something like changing a comment somewhere in hbase-server enough?
    label: documentation
17. Yes
18. Identical to HBASE-19486-20171219-1026.patch with a trivial comment change in hbase-server to trigger a QA run.
19. Fire QA run as requested by [~yuzhihong@gmail.com]
20. **body:** TODO for consistency: - setWriteBufferMaxLinger --> setWriteBufferMaxLingerMs - writeBufferMaxLinger --> writeBufferMaxLingerMs
    label: code-design
21. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 8s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue}

findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 12s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 28s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 59s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 30s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 59s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 46s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 12s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 36s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 26s{color} | {color:green} hbase-client: The patch generated 0 new + 24 unchanged - 6 fixed = 24 total (was 30) {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 2s{color} | {color:green} The patch hbase-server passed checkstyle {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} shadedjars {color} | {color:green} 4m 36s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 18m 47s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 45s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 43s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red}102m 44s{color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 33s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}145m 43s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12902824/HBASE-19486-20171219-1122-trigger-qa-run.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 98e60b4ff074 3.13.0-133-generic #182-Ubuntu SMP Tue Sep 19 15:49:21 UTC 2017 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 / 03e79b7994 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/10556/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10556/testReport/ | | modules | C: hbase-client hbase-server U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10556/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

22. The failure here was logged as {code} Failed to read test report file /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build/component/hbase-server/target/surefire-reports/TEST-org.apache.hadoop.hbase.master.TestDLSAsyncFSWAL.xml org.dom4j.DocumentException: Error on line 75 of document : XML document structures must start and end within the same entity. Nested exception: XML document structures must start and end within the same entity. at org.dom4j.io.SAXReader.read(SAXReader.java:482) at org.dom4j.io.SAXReader.read(SAXReader.java:343) at hudson.tasks.junit.SuiteResult.parse(SuiteResult.java:169) {code} As far as I can tell this has nothing to do with my patch. Correct?

23. **body:** Changes: - Naming consistency so everywhere writeBufferMaxLinger --> writeBufferMaxLingerMs - Some checkstyle messages fixed. NOTE: This still contains the extra 'NOP' change in hbase-server to trigger a full QA run.
    label: code-design
24. TestDLSAsyncFSWAL should not be related to your patch.
25. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 8s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 1s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 12s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 41s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 3s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 31s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 6m 28s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 56s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 15s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 31s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 12s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 12s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 32s{color} | {color:green} hbase-client: The patch generated 0 new + 24 unchanged - 6 fixed = 24 total (was 30) {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 15s{color} | {color:green} The patch hbase-server passed checkstyle {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} shadedjars {color} | {color:green} 5m 16s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 22m 32s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 59s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 3m 1s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red}111m 31s{color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 33s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}161m 11s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903068/HBASE-19486-20171220-1612-trigger-qa-run.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 8f8f6b5cb74a 3.13.0-133-generic #182-Ubuntu SMP Tue Sep 19 15:49:21 UTC 2017 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / 55fefd4b5a | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/10584/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10584/testReport/ | | modules | C: hbase-client hbase-server U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10584/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.
26. +1 Needs release note so folks have a chance of finding this new, useful addition. Thanks [~nielsbasjes]

27. +1 Nice! Good tests too. nit. Could you rename it to periodic flush or similar? If I recall, there is somewhere we use auto flush for a different meaning.
28. Maybe similar to {{HRegion.MEMSTORE_PERIODIC_FLUSH_INTERVAL}}?
29. [~jinghe] I used similar naming from what I know from Kafka which has a setting which effectively does something very similar: Look for {{linger.ms}} in this page: https://kafka.apache.org/documentation/ I personally think that having a similar naming compared to Kafka for this makes it easier for "Streaming people" to find. If you guys really prefer different naming then that's fine and I'll rename everything to make it consistent with the rest of HBase.
30. [~stack] I added a first draft release note to this issue.
31. RN is lovely. Appreciate your trying to align us w/ Kafka. Good. But [~jerryhe] has a point. autoflush in a client context particularly around BufferedMutator meant something else... See http://hbase.apache.org/1.2/apidocs/org/apache/hadoop/hbase/client/HTable.html#setAutoFlush-boolean- Probably better to name it something else sir.
32. Minor update to the patch. {code} - private transient Timer autoFlushTimer = null; + private Timer autoFlushTimer = null; {code} No need to give the impression of Serializable when it is not.
33. **body:** [~stack] Ok, I understand {{autoflush}} should not be used. Current name of the setting is {{writeBufferMaxLingerMs}} The only thing I currently named {{autoflush}} is the timer. I realize that this is inconsistent with the rest of this change (renaming that to {{writeBufferMaxLingerTimer}} is easy). But before I do this: What naming for this feature would you guys prefer? Something like {{write buffer periodic flush interval}} perhaps? Or is just getting rid of {{autoflush}} what you want?
label: code-design
34. I try to get a better feel for the word 'linger' to see if it fits better here. Still like {{write buffer periodic flush interva}} better at the moment to fit into hbase. Thanks for the effort to explain it.
35. What [~jerryhe] said.
36. Seems {{autoFlushCount}} is used by test only? If so, could we make {{autoFlush()}} be protected and then overrride it in test to get the count of auto flush. Should we cancel the previous task before invoking new timer task? {code} + autoFlushTimer = new Timer(true); // Create Timer running as Daemon. + autoFlushTimer.schedule(new TimerTask() { + @Override + public void run() { + BufferedMutatorImpl.this.autoFlush(); + } + }, writeBufferMaxLingerMs, writeBufferMaxLingerMs); {code}
37. Ok, thanks for clarifying. I'll get on it in the next few days.
38. Complete rename into what was discussed. I redesigned the algorithm because the previous one would flush too often in "full load" production situations.
39. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 0s{color} | {color:blue} Docker mode activated. {color} | | {color:red}-1{color} | {color:red} patch {color} | {color:red} 0m 4s{color} | {color:red} HBASE-19486 does not apply to master. Rebase required? Wrong Branch? See https://yetus.apache.org/documentation/0.6.0/precommit-patchnames for help. {color} | \\ \\ || Subsystem || Report/Notes || | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903525/HBASE-19486-20171223-1438-trigger-qa-run.patch | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10656/console | | Powered by | Apache Yetus 0.6.0 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 0s{color} | {color:blue} Docker mode activated. {color} | | {color:red}-1{color} | {color:red} patch {color} | {color:red} 0m 4s{color} | {color:red} HBASE-19486 does not apply to master. Rebase required? Wrong Branch? See https://yetus.apache.org/documentation/0.6.0/precommit-patchnames for help. {color} | \\ \\ || Subsystem || Report/Notes || | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903525/HBASE-19486-20171223-1438-trigger-qa-run.patch | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10657/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.
41. Patch needs rebasing. There are conflicts in BufferedMutatorImpl.java and TestAsyncProcess.java
42. Rebased
43. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 2m 25s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 22s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 14s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 57s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 21s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 24s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 40s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 11s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 17s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 56s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 56s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 22s{color} | {color:red} hbase-client: The patch generated 6 new + 24 unchanged - 3 fixed = 30 total (was 27) {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} shadedjars {color} | {color:green} 4m 3s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 17m 35s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 41s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 43s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green} 96m 1s{color} | {color:green} hbase-server in the patch passed. {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}139m 24s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903534/HBASE-19486-20171223-1728-trigger-qa-run.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux e2a32fc4ef78 4.4.0-43-generic #63-Ubuntu SMP Wed Oct 12 13:48:03 UTC 2016 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 / 2f25589422 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/10663/artifact/patchprocess/diff-checkstyle-hbase-client.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10663/testReport/ | | modules | C: hbase-client hbase-server U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10663/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

44. **body:** Checkstyle fixes
    label: code-design
45. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 8s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 12s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 31s{color} | {color:green} master passed {color} | |

{color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 32s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 6m 9s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 46s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 13s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 43s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 3s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 3s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 29s{color} | {color:red} hbase-client: The patch generated 2 new + 24 unchanged - 3 fixed = 26 total (was 27) {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} shadedjars {color} | {color:green} 4m 47s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 20m 15s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 54s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 40s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green}113m 58s{color} | {color:green} hbase-server in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 37s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}164m 55s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903544/HBASE-19486-20171223-2222-trigger-qa-run.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux e97cf831c920 3.13.0-129-generic #178-Ubuntu SMP Fri Aug 11 12:48:20 UTC 2017 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 / 2f25589422 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/10669/artifact/patchprocess/diff-checkstyle-hbase-client.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10669/testReport/ | | modules | C: hbase-client hbase-server U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10669/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

46. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 11s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 12s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 53s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 7s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 35s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 6m 18s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 51s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 13s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall

{color} | {color:green} 4m 56s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 3s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 3s{color} | {color:green} the patch passed {color} | | {color:red}-1{color} | {color:red} checkstyle {color} | {color:red} 0m 32s{color} | {color:red} hbase-client: The patch generated 2 new + 24 unchanged - 3 fixed = 26 total (was 27) {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} shadedjars {color} | {color:green} 4m 43s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 19m 26s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 49s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 49s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red}116m 40s{color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 34s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}170m 14s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903544/HBASE-19486-20171223-2222-trigger-qa-run.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux c50d36589383 3.13.0-133-generic #182-Ubuntu SMP Tue Sep 19 15:49:21 UTC 2017 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / 2f25589422 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | checkstyle | https://builds.apache.org/job/PreCommit-HBASE-Build/10668/artifact/patchprocess/diff-checkstyle-hbase-client.txt | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/10668/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10668/testReport/ | | modules | C: hbase-client hbase-server U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10668/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

47. {code} [ERROR] Failures: [ERROR] TestStochasticLoadBalancer2.testRegionReplicasOnLargeCluster:74->BalancerTestBase.testWithCluster:525->BalancerTestBase.testWithCluster:547->BalancerTestBase.assertClusterAsBalanced:207 {code} The above failure was not related to patch.
48. **body:** Clear. I'm having a hard time figuring out which checkstyle error made it fail because I cant open the Jenkins webui.
    label: code-design
49. **body:** Checkstyle fixes in test class
    label: code-design
50. If we pass the same parameters, the old Timer won't be ceased and an new timer will be created. {code} @Override public void setWriteBufferPeriodicFlush(long timeoutMs, long timerTickMs) { long originalTimeoutMs = this.writeBufferPeriodicFlushTimeoutMs; long originalTimerTickMs = this.writeBufferPeriodicFlushTimerTickMs; // Both parameters have minimal values. this.writeBufferPeriodicFlushTimeoutMs = Math.max(0, timeoutMs); this.writeBufferPeriodicFlushTimerTickMs = Math.max(MIN_WRITE_BUFFER_PERIODIC_FLUSH_TIMERTICK_MS, timerTickMs); // If something changed we stop the old Timer. if (this.writeBufferPeriodicFlushTimeoutMs != originalTimeoutMs || this.writeBufferPeriodicFlushTimerTickMs != originalTimerTickMs) { if (writeBufferPeriodicFlushTimer != null) { writeBufferPeriodicFlushTimer.cancel(); writeBufferPeriodicFlushTimer = null; } } // If we have the need for a new timer we start it if (this.writeBufferPeriodicFlushTimeoutMs > 0) { writeBufferPeriodicFlushTimer = new Timer(true); // Create Timer running as Daemon. writeBufferPeriodicFlushTimer.schedule(new TimerTask() { @Override public void run() { BufferedMutatorImpl.this.timerCallbackForWriteBufferPeriodicFlush(); } }, writeBufferPeriodicFlushTimerTickMs, writeBufferPeriodicFlushTimerTickMs); } } {code}
51. Fixed double timer problem.

52. | (/) *{color:green}+1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 11s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 32s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 23s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 33s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 44s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 22s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 5m 13s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 29s{color} | {color:green} hbase-client: The patch generated 0 new + 24 unchanged - 3 fixed = 24 total (was 27) {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} shadedjars {color} | {color:green} 5m 11s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 22m 51s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 22s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 57s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 9s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 44m 49s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903571/HBASE-19486-20171224-1602.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux b6f5aa26b8b2 3.13.0-129-generic #178-Ubuntu SMP Fri Aug 11 12:48:20 UTC 2017 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 / c24cf2d55e | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10683/testReport/ | | modules | C: hbase-client U: hbase-client | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10683/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.
53. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 1m 55s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 21s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 52s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 7s{color} | {color:green}

master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 33s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 6m 11s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 52s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 13s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 51s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 9s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 9s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 29s{color} | {color:green} hbase-client: The patch generated 0 new + 24 unchanged - 3 fixed = 24 total (was 27) {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 5s{color} | {color:green} The patch hbase-server passed checkstyle {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} shadedjars {color} | {color:green} 4m 39s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 20m 2s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 47s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 44s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:red}-1{color} | {color:red} unit {color} | {color:red}116m 2s{color} | {color:red} hbase-server in the patch failed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 34s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}163m 32s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903554/HBASE-19486-20171224-1101-trigger-qa-run.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux aa2a0405e3b3 3.13.0-133-generic #182-Ubuntu SMP Tue Sep 19 15:49:21 UTC 2017 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / c24cf2d55e | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | unit | https://builds.apache.org/job/PreCommit-HBASE-Build/10672/artifact/patchprocess/patch-unit-hbase-server.txt | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10672/testReport/ | | modules | C: hbase-client hbase-server U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10672/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

54. Jenkins seems to be having problems. This last Hadoop QA post belongs to an older patch.
55. The patch you submit to trigger full QA doesn't fix the double timer problem.
56. Hi [~chia7712] You are right; the last Jenkins report does not solve that issue. The reason is that Jenkins has been having a lot of issues these last few days (see http://status.apache.org/) The last message from Jenkins is from a patch that is 5 hours older than the one before that. The patch I submitted that does solve this issue does not do a full run (https://issues.apache.org/jira/secure/attachment/12903571/HBASE-19486-20171224-1602.patch). Niels
57. I believe the failure is unrelated to your patch. :) However, it would be better to use the correct patch to trigger the full run. I attach the patch with rebase and trivial changes in hbase-server. Will commit it if all green.
58. BTW, would you please prepare the patch for branch-1?
59. | (/) *{color:green}+1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 9s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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 3 new or modified test files. {color} | || || || || {color:brown} master Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 12s{color} | {color:blue} Maven dependency ordering for branch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 33s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 0s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 30s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 6m 4s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 46s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:blue}0{color} | {color:blue} mvndep {color} | {color:blue} 0m 13s{color} | {color:blue} Maven dependency ordering for patch {color} | | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 33s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 1m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 1m 0s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 28s{color} | {color:green} hbase-client: The patch generated 0 new + 24 unchanged - 3 fixed = 24 total (was 27) {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 1m 2s{color} | {color:green} The patch hbase-server passed checkstyle {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} shadedjars {color} | {color:green} 4m 44s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 20m 27s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 52s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 54s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} unit {color} | {color:green}106m 17s{color} | {color:green} hbase-server in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 36s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black}151m 26s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12903819/HBASE-19486.v0.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux e3e7fd5ddb14 3.13.0-129-generic #178-Ubuntu SMP Fri Aug 11 12:48:20 UTC 2017 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 / 6b39062e86 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10751/testReport/ | | modules | C: hbase-client hbase-server U: . | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10751/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

60. [~chia7712] Yes, if you guys fully agree with this implementation I will prepare a backport against branch-1
61. bq. Yes, if you guys fully agree with this implementation I will prepare a backport against branch-1 +1
62. Seems the patch doesn't cease the timer in BF#close(). Would you please fix it?
63. [~chia7712] The BFImpl#close does a call to {{disableWriteBufferPeriodicFlush();}} which cancels the running timer if present. So to me this seem "not a problem" as far as I can see. I'm preparing the branch-1 patch now.
64. {quote} The BFImpl#close does a call to disableWriteBufferPeriodicFlush(); which cancels the running timer if present. So to me this seem "not a problem" as far as I can see. {quote} Thanks! I missed that.
65. The backport of the patch to branch-1. I have tried to keep this set of changes "as close as possible" to the original patch against master. Most notable differences # because of Java 1.7 the interface cannot have default implementations. # I have also included a few parts (mostly tests) from the master branch to test everything in an as similar as possible way.
66. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 0s{color} | {color:blue} Docker mode activated. {color} | | {color:red}-1{color} | {color:red} patch {color} | {color:red} 0m 6s{color} |

{color:red} HBASE-19486 does not apply to branch-1. Rebase required? Wrong Branch? See https://yetus.apache.org/documentation/0.6.0/precommit-patchnames for help. {color} | \\ \\ || Subsystem || Report/Notes || | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12904010/HBASE-19486-branch-1.v0.patch | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10789/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

67. Rebased
68. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 0s{color} | {color:blue} Docker mode activated. {color} | | {color:red}-1{color} | {color:red} docker {color} | {color:red} 13m 26s{color} | {color:red} Docker failed to build yetus/hbase:36a7029. {color} | \\ \\ || Subsystem || Report/Notes || | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12904011/HBASE-19486-branch-1.v1.patch | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10790/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.
69. bq. because of Java 1.7 the interface cannot have default implementations. Seems it break our BC if we backport this feature to branch-1. Let me commit it to master and branch-2. We can open another issue to discuss how to backport this feature to branch-1 without breaking the BC.
70. Thanks for the nice feature. [~nielsbasjes]
71. FAILURE: Integrated in Jenkins build HBase-Trunk_matrix #4307 (See [https://builds.apache.org/job/HBase-Trunk_matrix/4307/]) HBASE-19486 Periodically ensure records are not buffered too long by (chia7712: rev 5a1c36f70ac52e6f4e85f11ea0602d46b4861ac0) * (edit) hbase-client/src/test/java/org/apache/hadoop/hbase/client/TestAsyncProcess.java * (edit) hbase-client/src/main/java/org/apache/hadoop/hbase/client/BufferedMutatorParams.java * (edit) hbase-client/src/test/java/org/apache/hadoop/hbase/client/TestBufferedMutatorParams.java * (edit) hbase-client/src/main/java/org/apache/hadoop/hbase/client/ConnectionImplementation.java * (edit) hbase-client/src/main/java/org/apache/hadoop/hbase/client/BufferedMutator.java * (edit) hbase-client/src/main/java/org/apache/hadoop/hbase/client/ConnectionConfiguration.java * (edit) hbase-client/src/main/java/org/apache/hadoop/hbase/client/BufferedMutatorImpl.java
72. The BM is assumed to be thread-safe. Would you please attach an addendum to make timer setter/getter be thread-safe?
73. Yes, will do.
74. Made all WriteBufferPeriodicFlush related operations threadsafe by using AtomicLong instead of long and making the method that sets everything (setWriteBufferPeriodicFlush) to synchronized. [~chia7712] Please verify to check if I did it correctly/missed anything.
75. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 0m 9s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 44s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 24s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 31s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 16s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 19s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 44s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 21s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 21s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 28s{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} shadedjars {color} | {color:green} 4m 52s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 20m 23s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 20s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 49s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 8s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 40m 17s{color} | {color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12904100/HBASE-19486.20171231-105839-addendum.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux 0395b4d63311 3.13.0-133-generic #182-Ubuntu SMP Tue Sep 19 15:49:21 UTC 2017 x86_64 GNU/Linux | | Build tool | maven | | Personality | /home/jenkins/jenkins-slave/workspace/PreCommit-HBASE-Build@2/component/dev-support/hbase-personality.sh | | git revision | master / 0cd6050d09 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10816/testReport/ | | modules | C: hbase-client U: hbase-client | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10816/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

76. Could you make the fields be final? Otherwise LGTM
77. Made the two AtomicLong fields final
78. | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | {color:blue} reexec {color} | {color:blue} 1m 53s{color} | {color:blue} Docker mode activated. {color} | || || || || {color:brown} Prechecks {color} || | {color:blue}0{color} | {color:blue} findbugs {color} | {color:blue} 0m 0s{color} | {color:blue} Findbugs executables are not available. {color} | | {color:green}+1{color} | {color:green} hbaseanti {color} | {color:green} 0m 0s{color} | {color:green} Patch does not have any anti-patterns. {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:brown} master Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 34s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 19s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 26s{color} | {color:green} master passed {color} | | {color:green}+1{color} | {color:green} shadedjars {color} | {color:green} 5m 5s{color} | {color:green} branch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 19s{color} | {color:green} master passed {color} | || || || || {color:brown} Patch Compile Tests {color} || | {color:green}+1{color} | {color:green} mvninstall {color} | {color:green} 4m 34s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} compile {color} | {color:green} 0m 19s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} javac {color} | {color:green} 0m 19s{color} | {color:green} the patch passed {color} | | {color:green}+1{color} | {color:green} checkstyle {color} | {color:green} 0m 27s{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} shadedjars {color} | {color:green} 4m 34s{color} | {color:green} patch has no errors when building our shaded downstream artifacts. {color} | | {color:green}+1{color} | {color:green} hadoopcheck {color} | {color:green} 19m 8s{color} | {color:green} Patch does not cause any errors with Hadoop 2.6.5 2.7.4 or 3.0.0. {color} | | {color:green}+1{color} | {color:green} javadoc {color} | {color:green} 0m 19s{color} | {color:green} the patch passed {color} | || || || || {color:brown} Other Tests {color} || | {color:green}+1{color} | {color:green} unit {color} | {color:green} 2m 41s{color} | {color:green} hbase-client in the patch passed. {color} | | {color:green}+1{color} | {color:green} asflicense {color} | {color:green} 0m 9s{color} | {color:green} The patch does not generate ASF License warnings. {color} | | {color:black}{color} | {color:black} {color} | {color:black} 39m 58s{color} |

{color:black} {color} | \\ \\ || Subsystem || Report/Notes || | Docker | Client=17.05.0-ce Server=17.05.0-ce Image:yetus/hbase:eee3b01 | | JIRA Issue | HBASE-19486 | | JIRA Patch URL | https://issues.apache.org/jira/secure/attachment/12904169/HBASE-19486.20180102-081903-addendum.patch | | Optional Tests | asflicense javac javadoc unit findbugs shadedjars hadoopcheck hbaseanti checkstyle compile | | uname | Linux fcdbc7a21572 3.13.0-129-generic #178-Ubuntu SMP Fri Aug 11 12:48:20 UTC 2017 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 / 6708d54478 | | maven | version: Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z) | | Default Java | 1.8.0_151 | | Test Results | https://builds.apache.org/job/PreCommit-HBASE-Build/10841/testReport/ | | modules | C: hbase-client U: hbase-client | | Console output | https://builds.apache.org/job/PreCommit-HBASE-Build/10841/console | | Powered by | Apache Yetus 0.6.0 http://yetus.apache.org | This message was automatically generated.

79. Thanks for the contributions. [~nielsbasjes]
80. FAILURE: Integrated in Jenkins build HBase-Trunk_matrix #4330 (See [https://builds.apache.org/job/HBase-Trunk_matrix/4330/]) HBASE-19486: Ensure threadsafe WriteBufferPeriodicFlush operations (chia7712: rev a6081d30f930d9599f7d52ab440b3205c7f2a7bf) * (edit) hbase-client/src/main/java/org/apache/hadoop/hbase/client/BufferedMutatorImpl.java