package com.twitter.search.earlybird.partition;

import java.io.IOException;

import java.util.Optional;

import java.util.concurrent.TimeUnit;

import com.google.common.base.Preconditions;

import com.google.common.base.Stopwatch;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.indexing.thriftjava.ThriftVersionedEvents;

import com.twitter.search.common.metrics.SearchTimer;

import com.twitter.search.common.util.io.dl.DLRecordTimestampUtil;

import com.twitter.search.common.util.io.recordreader.RecordReader;

import com.twitter.search.earlybird.exception.CriticalExceptionHandler;

import com.twitter.search.earlybird.segment.SegmentDataReaderSet;

/\*\*

\* Indexes all updates for a complete segment at startup.

\*/

public class SimpleUpdateIndexer {

private static final Logger LOG = LoggerFactory.getLogger(SimpleUpdateIndexer.class);

private final SegmentDataReaderSet readerSet;

private final SearchIndexingMetricSet partitionIndexingMetricSet;

private final InstrumentedQueue<ThriftVersionedEvents> retryQueue;

private final CriticalExceptionHandler criticalExceptionHandler;

public SimpleUpdateIndexer(SegmentDataReaderSet readerSet,

SearchIndexingMetricSet partitionIndexingMetricSet,

InstrumentedQueue<ThriftVersionedEvents> retryQueue,

CriticalExceptionHandler criticalExceptionHandler) {

this.readerSet = readerSet;

this.partitionIndexingMetricSet = partitionIndexingMetricSet;

this.retryQueue = retryQueue;

this.criticalExceptionHandler = criticalExceptionHandler;

}

/\*\*

\* Indexes all updates for the given segment.

\*/

public void indexAllUpdates(SegmentInfo segmentInfo) {

Preconditions.checkState(

segmentInfo.isEnabled() && segmentInfo.isComplete() && !segmentInfo.isIndexing());

try {

readerSet.attachUpdateReaders(segmentInfo);

} catch (IOException e) {

throw new RuntimeException("Could not attach readers for segment: " + segmentInfo, e);

}

RecordReader<ThriftVersionedEvents> reader =

readerSet.getUpdateEventsReaderForSegment(segmentInfo);

if (reader == null) {

return;

}

LOG.info("Got updates reader (starting timestamp = {}) for segment {}: {}",

DLRecordTimestampUtil.recordIDToTimestamp(reader.getOffset()),

segmentInfo.getSegmentName(),

reader);

// The segment is complete (we check this in indexAllUpdates()), so we can safely get

// the smallest and largest tweet IDs in this segment.

long lowestTweetId = segmentInfo.getIndexSegment().getLowestTweetId();

long highestTweetId = segmentInfo.getIndexSegment().getHighestTweetId();

Preconditions.checkArgument(

lowestTweetId > 0,

"Could not get the lowest tweet ID in segment " + segmentInfo.getSegmentName());

Preconditions.checkArgument(

highestTweetId > 0,

"Could not get the highest tweet ID in segment " + segmentInfo.getSegmentName());

SegmentWriter segmentWriter =

new SegmentWriter(segmentInfo, partitionIndexingMetricSet.updateFreshness);

LOG.info("Starting to index updates for segment: {}", segmentInfo.getSegmentName());

Stopwatch stopwatch = Stopwatch.createStarted();

while (!Thread.currentThread().isInterrupted() && !reader.isCaughtUp()) {

applyUpdate(segmentInfo, reader, segmentWriter, lowestTweetId, highestTweetId);

}

LOG.info("Finished indexing updates for segment {} in {} seconds.",

segmentInfo.getSegmentName(),

stopwatch.elapsed(TimeUnit.SECONDS));

}

private void applyUpdate(SegmentInfo segmentInfo,

RecordReader<ThriftVersionedEvents> reader,

SegmentWriter segmentWriter,

long lowestTweetId,

long highestTweetId) {

ThriftVersionedEvents update;

try {

update = reader.readNext();

} catch (IOException e) {

LOG.error("Exception while reading update for segment: " + segmentInfo.getSegmentName(), e);

criticalExceptionHandler.handle(this, e);

return;

}

if (update == null) {

LOG.warn("Update is not available but reader was not caught up. Segment: {}",

segmentInfo.getSegmentName());

return;

}

try {

// If the indexer put this update in the wrong timeslice, add it to the retry queue, and

// let PartitionIndexer retry it (it has logic to apply it to the correct segment).

if ((update.getId() < lowestTweetId) || (update.getId() > highestTweetId)) {

retryQueue.add(update);

return;

}

// At this point, we are updating a segment that has every tweet it will ever have,

// (the segment is complete), so there is no point queueing an update to retry it.

SearchTimer timer = partitionIndexingMetricSet.updateStats.startNewTimer();

segmentWriter.indexThriftVersionedEvents(update);

partitionIndexingMetricSet.updateStats.stopTimerAndIncrement(timer);

updateUpdatesStreamTimestamp(segmentInfo);

} catch (IOException e) {

LOG.error("Exception while indexing updates for segment: " + segmentInfo.getSegmentName(), e);

criticalExceptionHandler.handle(this, e);

}

}

private void updateUpdatesStreamTimestamp(SegmentInfo segmentInfo) {

Optional<Long> offset = readerSet.getUpdateEventsStreamOffsetForSegment(segmentInfo);

if (!offset.isPresent()) {

LOG.info("Unable to get updates stream offset for segment: {}", segmentInfo.getSegmentName());

} else {

long offsetTimeMillis = DLRecordTimestampUtil.recordIDToTimestamp(offset.get());

segmentInfo.setUpdatesStreamOffsetTimestamp(offsetTimeMillis);

}

}

}