package com.twitter.search.earlybird.partition;

import java.io.IOException;

import java.util.concurrent.TimeUnit;

import javax.annotation.Nullable;

import com.google.common.base.Stopwatch;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

import com.twitter.search.earlybird.common.config.EarlybirdConfig;

import com.twitter.search.earlybird.document.TweetDocument;

import com.twitter.search.earlybird.index.EarlybirdSegment;

/\*\*

\* SimpleSegmentIndex indexes all Tweets for a \*complete\* segment. It does not index any updates or

\* deletes.

\*/

public class SimpleSegmentIndexer {

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

/\*\*

\* If not null, this segment is appended at the end after indexing finishes.

\*/

@Nullable

private final SegmentInfo segmentToAppend;

private final RecordReader<TweetDocument> tweetReader;

private final SearchIndexingMetricSet partitionIndexingMetricSet;

// Segment we are indexing.

private EarlybirdSegment indexingSegment;

// Total number of statuses indexed in this segment.

private long segmentSize = 0;

public SimpleSegmentIndexer(

RecordReader<TweetDocument> tweetReader,

SearchIndexingMetricSet partitionIndexingMetricSet) {

this(tweetReader, partitionIndexingMetricSet, null);

}

public SimpleSegmentIndexer(RecordReader<TweetDocument> tweetReader,

SearchIndexingMetricSet partitionIndexingMetricSet,

@Nullable SegmentInfo segmentToAppend) {

this.tweetReader = tweetReader;

this.segmentToAppend = segmentToAppend;

this.partitionIndexingMetricSet = partitionIndexingMetricSet;

}

private boolean shouldIndexSegment(SegmentInfo segmentInfo) {

if (!segmentInfo.isEnabled()) {

return false;

}

if (segmentToAppend != null) {

return true;

}

return !segmentInfo.isComplete()

&& !segmentInfo.isIndexing()

&& !segmentInfo.getSyncInfo().isLoaded();

}

/\*\*

\* Indexes all tweets for a complete segment.

\*/

public boolean indexSegment(SegmentInfo segmentInfo) {

LOG.info("Indexing segment " + segmentInfo.getSegmentName());

if (!shouldIndexSegment(segmentInfo)) {

return false;

}

// If we're starting to index, we're not complete, will become complete if we

// were successful here.

segmentInfo.setComplete(false);

try {

segmentInfo.setIndexing(true);

indexingSegment = segmentInfo.getIndexSegment();

// if we're updating the segment, then we'll index only the new available days

// and then append the lucene index from the old segment

// If segmentToAppend is not null, it means we are updating a segment.

if (indexingSegment.tryToLoadExistingIndex()) {

segmentInfo.getSyncInfo().setLoaded(true);

LOG.info("Loaded existing index for " + segmentInfo + ", not indexing.");

} else {

indexingLoop();

if (segmentToAppend != null) {

indexingSegment.append(segmentToAppend.getIndexSegment());

}

}

segmentInfo.setIndexing(false);

segmentInfo.setComplete(true);

segmentInfo.setWasIndexed(true);

LOG.info("Successfully indexed segment " + segmentInfo.getSegmentName());

return true;

} catch (Exception e) {

LOG.error("Exception while indexing IndexSegment " + segmentInfo

+ " after " + indexingSegment.getIndexStats().getStatusCount() + " documents.", e);

partitionIndexingMetricSet.simpleSegmentIndexerExceptionCounter.increment();

LOG.warn("Failed to load a new day into full archive. Cleaning up segment: "

+ indexingSegment.getSegmentName());

// Clean up the lucene dir if it exists. Earlybird will retry loading the new day again later.

if (!segmentInfo.deleteLocalIndexedSegmentDirectoryImmediately()) {

LOG.error("Failed to clean up index segment folder after indexing failures.");

}

return false;

} finally {

if (tweetReader != null) {

tweetReader.stop();

}

segmentInfo.setIndexing(false);

}

}

// Indexes a document if available. Returns true if index was updated.

protected boolean indexDocument(TweetDocument tweetDocument) throws IOException {

if (tweetDocument == null) {

return false;

}

SearchTimer timer = partitionIndexingMetricSet.statusStats.startNewTimer();

indexingSegment.addDocument(tweetDocument);

partitionIndexingMetricSet.statusStats.stopTimerAndIncrement(timer);

segmentSize++;

return true;

}

/\*\*

\* Indexes all tweets for this segment, until no more tweets are available.

\*

\* @throws InterruptedException If the thread is interrupted while indexing tweets.

\* @throws IOException If there's a problem reading or indexing tweets.

\*/

public void indexingLoop() throws InterruptedException, IOException {

Stopwatch stopwatch = Stopwatch.createStarted();

Stopwatch readingStopwatch = Stopwatch.createUnstarted();

Stopwatch indexingStopwatch = Stopwatch.createUnstarted();

int indexedDocumentsCount = 0;

SearchLongGauge timeToIndexSegment = SearchLongGauge.export("time\_to\_index\_segment");

timeToIndexSegment.set(0);

if (tweetReader != null) {

while (!tweetReader.isExhausted() && !Thread.currentThread().isInterrupted()) {

readingStopwatch.start();

TweetDocument tweetDocument = tweetReader.readNext();

readingStopwatch.stop();

indexingStopwatch.start();

boolean documentIndexed = indexDocument(tweetDocument);

indexingStopwatch.stop();

if (!documentIndexed) {

// No documents waiting to be indexed. Take a nap.

Thread.sleep(10);

} else {

indexedDocumentsCount++;

}

if (segmentSize >= EarlybirdConfig.getMaxSegmentSize()) {

LOG.error("Reached max segment size " + segmentSize + ", stopping indexer");

partitionIndexingMetricSet.maxSegmentSizeReachedCounter.increment();

tweetReader.stop();

break;

}

}

}

timeToIndexSegment.set(stopwatch.elapsed(TimeUnit.MILLISECONDS));

LOG.info("SimpleSegmentIndexer finished: {}. Documents: {}",

indexingSegment.getSegmentName(), indexedDocumentsCount);

LOG.info("Time taken: {}, Reading time: {}, Indexing time: {}",

stopwatch, readingStopwatch, indexingStopwatch);

LOG.info("Total Memory: {}, Free Memory: {}",

Runtime.getRuntime().totalMemory(), Runtime.getRuntime().freeMemory());

}

}