package com.twitter.search.earlybird.partition;

import java.io.IOException;

import java.time.Duration;

import com.google.common.annotations.VisibleForTesting;

import org.apache.kafka.clients.consumer.ConsumerRecord;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.common.util.Clock;

import com.twitter.common\_internal.text.version.PenguinVersion;

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

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

import com.twitter.search.common.partitioning.snowflakeparser.SnowflakeIdParser;

import com.twitter.search.common.schema.thriftjava.ThriftIndexingEvent;

import com.twitter.search.common.schema.thriftjava.ThriftIndexingEventType;

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

/\*\*

\* PartitionWriter writes Tweet events and Tweet update events to an Earlybird index. It is

\* responsible for creating new segments, adding Tweets to the correct segment, and applying updates

\* to the correct segment.

\*/

public class PartitionWriter {

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

private static final String STATS\_PREFIX = "partition\_writer\_";

private static final SearchRateCounter MISSING\_PENGUIN\_VERSION =

SearchRateCounter.export(STATS\_PREFIX + "missing\_penguin\_version");

private static final Duration CAUGHT\_UP\_FRESHNESS = Duration.ofSeconds(5);

private static final SearchRateCounter EVENTS\_CONSUMED =

SearchRateCounter.export(STATS\_PREFIX + "events\_consumed");

private final PenguinVersion penguinVersion;

private final TweetUpdateHandler updateHandler;

private final TweetCreateHandler createHandler;

private final Clock clock;

private final CriticalExceptionHandler criticalExceptionHandler;

public PartitionWriter(

TweetCreateHandler tweetCreateHandler,

TweetUpdateHandler tweetUpdateHandler,

CriticalExceptionHandler criticalExceptionHandler,

PenguinVersion penguinVersion,

Clock clock

) {

LOG.info("Creating PartitionWriter.");

this.createHandler = tweetCreateHandler;

this.updateHandler = tweetUpdateHandler;

this.criticalExceptionHandler = criticalExceptionHandler;

this.penguinVersion = penguinVersion;

this.clock = clock;

}

/\*\*

\* Index a batch of TVE records.

\*/

public boolean indexBatch(Iterable<ConsumerRecord<Long, ThriftVersionedEvents>> records)

throws Exception {

long minTweetAge = Long.MAX\_VALUE;

for (ConsumerRecord<Long, ThriftVersionedEvents> record : records) {

ThriftVersionedEvents tve = record.value();

indexTVE(tve);

EVENTS\_CONSUMED.increment();

long tweetAgeInMs = SnowflakeIdParser.getTweetAgeInMs(clock.nowMillis(), tve.getId());

minTweetAge = Math.min(tweetAgeInMs, minTweetAge);

}

return minTweetAge < CAUGHT\_UP\_FRESHNESS.toMillis();

}

/\*\*

\* Index a ThriftVersionedEvents struct.

\*/

@VisibleForTesting

public void indexTVE(ThriftVersionedEvents tve) throws IOException {

ThriftIndexingEvent tie = tve.getVersionedEvents().get(penguinVersion.getByteValue());

if (tie == null) {

LOG.error("Could not find a ThriftIndexingEvent for PenguinVersion {} in "

+ "ThriftVersionedEvents: {}", penguinVersion, tve);

MISSING\_PENGUIN\_VERSION.increment();

return;

}

// An `INSERT` event is used for new Tweets. These are generated from Tweet Create Events from

// TweetyPie.

if (tie.getEventType() == ThriftIndexingEventType.INSERT) {

createHandler.handleTweetCreate(tve);

updateHandler.retryPendingUpdates(tve.getId());

} else {

updateHandler.handleTweetUpdate(tve, false);

}

}

public void prepareAfterStartingWithIndex(long maxIndexedTweetId) {

createHandler.prepareAfterStartingWithIndex(maxIndexedTweetId);

}

void logState() {

LOG.info("PartitionWriter state:");

LOG.info(String.format(" Events indexed: %,d", EVENTS\_CONSUMED.getCount()));

createHandler.logState();

updateHandler.logState();

}

}