package com.twitter.search.earlybird.partition;

import java.util.Date;

import com.google.common.annotations.VisibleForTesting;

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

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

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

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

import com.twitter.search.common.util.io.kafka.CompactThriftDeserializer;

import com.twitter.search.common.util.io.kafka.FinagleKafkaClientUtils;

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

import com.twitter.search.earlybird.common.userupdates.UserUpdate;

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

public class UserUpdatesStreamIndexer extends SimpleStreamIndexer<Long, AntisocialUserUpdate> {

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

private static final SearchCounter NUM\_CORRUPT\_DATA\_ERRORS =

SearchCounter.export("num\_user\_updates\_kafka\_consumer\_corrupt\_data\_errors");

protected static String kafkaClientId = "";

private final SegmentManager segmentManager;

private final SearchIndexingMetricSet searchIndexingMetricSet;

public UserUpdatesStreamIndexer(KafkaConsumer<Long, AntisocialUserUpdate> kafkaConsumer,

String topic,

SearchIndexingMetricSet searchIndexingMetricSet,

SegmentManager segmentManager)

throws MissingKafkaTopicException {

super(kafkaConsumer, topic);

this.segmentManager = segmentManager;

this.searchIndexingMetricSet = searchIndexingMetricSet;

indexingSuccesses = SearchRateCounter.export("user\_update\_indexing\_successes");

indexingFailures = SearchRateCounter.export("user\_update\_indexing\_failures");

}

/\*\*

\* Provides user updates kafka consumer to EarlybirdWireModule.

\* @return

\*/

public static KafkaConsumer<Long, AntisocialUserUpdate> provideKafkaConsumer() {

return FinagleKafkaClientUtils.newKafkaConsumerForAssigning(

EarlybirdProperty.KAFKA\_PATH.get(),

new CompactThriftDeserializer<>(AntisocialUserUpdate.class),

kafkaClientId,

MAX\_POLL\_RECORDS);

}

UserUpdate convertToUserInfoUpdate(AntisocialUserUpdate update) {

return new UserUpdate(

update.getUserID(),

update.getType(),

update.isValue() ? 1 : 0,

new Date(update.getUpdatedAt()));

}

@VisibleForTesting

protected void validateAndIndexRecord(ConsumerRecord<Long, AntisocialUserUpdate> record) {

AntisocialUserUpdate update = record.value();

if (update == null) {

LOG.warn("null value returned from poll");

return;

}

if (update.getType() == null) {

LOG.error("User update does not have type set: " + update);

NUM\_CORRUPT\_DATA\_ERRORS.increment();

return;

}

SearchTimer timer = searchIndexingMetricSet.userUpdateIndexingStats.startNewTimer();

boolean isUpdateIndexed = segmentManager.indexUserUpdate(

convertToUserInfoUpdate(update));

searchIndexingMetricSet.userUpdateIndexingStats.stopTimerAndIncrement(timer);

if (isUpdateIndexed) {

indexingSuccesses.increment();

} else {

indexingFailures.increment();

}

}

}