package com.twitter.search.ingester.pipeline.twitter.kafka;

import javax.naming.NamingException;

import org.apache.commons.pipeline.StageException;

import org.apache.commons.pipeline.validation.ConsumedTypes;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.search.common.debug.DebugEventUtil;

import com.twitter.search.common.debug.thriftjava.DebugEvents;

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

import com.twitter.search.ingester.model.IngesterThriftVersionedEvents;

import com.twitter.search.ingester.pipeline.util.PipelineStageException;

/\*\*

\* Kafka producer stage to write tweet indexing data as {@code ThriftVersionedEvents}. This stage

\* also handles extra debug event processing.

\*/

@ConsumedTypes(IngesterThriftVersionedEvents.class)

public class TweetThriftVersionedEventsKafkaProducerStage extends KafkaProducerStage

<IngesterThriftVersionedEvents> {

private static final int PROCESSING\_LATENCY\_THRESHOLD\_FOR\_UPDATES\_MILLIS = 30000;

private static final Logger LOG =

LoggerFactory.getLogger(TweetThriftVersionedEventsKafkaProducerStage.class);

private long processedTweetCount = 0;

private SearchLongGauge kafkaProducerLag;

private int debugEventLogPeriod = -1;

public TweetThriftVersionedEventsKafkaProducerStage(String kafkaTopic, String clientId,

String clusterPath) {

super(kafkaTopic, clientId, clusterPath);

}

public TweetThriftVersionedEventsKafkaProducerStage() {

super();

}

@Override

protected void initStats() {

super.initStats();

setupCommonStats();

}

@Override

protected void innerSetupStats() {

super.innerSetupStats();

setupCommonStats();

}

private void setupCommonStats() {

kafkaProducerLag = SearchLongGauge.export(

getStageNamePrefix() + "\_kafka\_producer\_lag\_millis");

}

@Override

protected void innerSetup() throws PipelineStageException, NamingException {

super.innerSetup();

}

@Override

protected void doInnerPreprocess() throws StageException, NamingException {

super.doInnerPreprocess();

commonInnerSetup();

}

private void commonInnerSetup() {

setProcessingLatencyThresholdMillis(PROCESSING\_LATENCY\_THRESHOLD\_FOR\_UPDATES\_MILLIS);

}

@Override

public void innerProcess(Object obj) throws StageException {

if (!(obj instanceof IngesterThriftVersionedEvents)) {

throw new StageException(this, "Object is not IngesterThriftVersionedEvents: " + obj);

}

IngesterThriftVersionedEvents events = (IngesterThriftVersionedEvents) obj;

innerRunFinalStageOfBranchV2(events);

}

@Override

protected void innerRunFinalStageOfBranchV2(IngesterThriftVersionedEvents events) {

if ((debugEventLogPeriod > 0)

&& (processedTweetCount % debugEventLogPeriod == 0)

&& (events.getDebugEvents() != null)) {

LOG.info("DebugEvents for tweet {}: {}",

events.getTweetId(), DebugEventUtil.debugEventsToString(events.getDebugEvents()));

}

processedTweetCount++;

DebugEvents debugEvents = events.getDebugEvents();

if ((debugEvents != null) && debugEvents.isSetProcessingStartedAt()) {

kafkaProducerLag.set(

clock.nowMillis() - debugEvents.getProcessingStartedAt().getEventTimestampMillis());

}

super.tryToSendEventsToKafka(events);

}

@SuppressWarnings("unused") // set from pipeline config

public void setDebugEventLogPeriod(int debugEventLogPeriod) {

this.debugEventLogPeriod = debugEventLogPeriod;

}

}