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

import java.io.IOException;

import java.util.List;

import java.util.Optional;

import javax.naming.NamingException;

import com.google.common.base.Preconditions;

import org.apache.commons.lang.StringUtils;

import org.apache.commons.pipeline.StageException;

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

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import com.twitter.search.common.converter.earlybird.BasicIndexingConverter;

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

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

import com.twitter.search.common.schema.base.Schema;

import com.twitter.search.common.schema.earlybird.EarlybirdSchemaCreateTool;

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

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

@ConsumedTypes(IngesterTwitterMessage.class)

@ProducesConsumed

public class ConvertMessageToThriftStage extends TwitterBaseStage

<IngesterTwitterMessage, IngesterTwitterMessage> {

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

private List<PenguinVersion> penguinVersionList;

private String thriftVersionedEventsBranchName;

private FieldStatExporter fieldStatExporter;

private BasicIndexingConverter messageConverter;

private SearchCounter twitterMessageToTveErrorCount;

@Override

public void initStats() {

super.initStats();

twitterMessageToTveErrorCount = SearchCounter.export(

getStageNamePrefix() + "\_ingester\_convert\_twitter\_message\_to\_tve\_error\_count");

}

@Override

protected void doInnerPreprocess() throws StageException, NamingException {

Schema schema;

try {

schema = EarlybirdSchemaCreateTool.buildSchema(Preconditions.checkNotNull(earlybirdCluster));

} catch (Schema.SchemaValidationException e) {

throw new StageException(this, e);

}

penguinVersionList = wireModule.getPenguinVersions();

Preconditions.checkState(StringUtils.isNotBlank(thriftVersionedEventsBranchName));

messageConverter = new BasicIndexingConverter(schema, earlybirdCluster);

fieldStatExporter = new FieldStatExporter("unsorted\_tweets", schema, penguinVersionList);

}

@Override

public void innerProcess(Object obj) throws StageException {

if (!(obj instanceof IngesterTwitterMessage)) {

throw new StageException(this, "Object is not an IngesterTwitterMessage instance: " + obj);

}

penguinVersionList = wireModule.getCurrentlyEnabledPenguinVersions();

fieldStatExporter.updatePenguinVersions(penguinVersionList);

IngesterTwitterMessage message = IngesterTwitterMessage.class.cast(obj);

Optional<IngesterThriftVersionedEvents> maybeEvents = buildVersionedEvents(message);

if (maybeEvents.isPresent()) {

IngesterThriftVersionedEvents events = maybeEvents.get();

fieldStatExporter.addFieldStats(events);

emitToBranchAndCount(thriftVersionedEventsBranchName, events);

}

emitAndCount(message);

}

/\*\*

\* Method that converts a TwitterMessage to a ThriftVersionedEvents.

\*

\* @param twitterMessage An IngesterThriftVersionedEvents instance to be converted.

\* @return The corresponding ThriftVersionedEvents.

\*/

private Optional<IngesterThriftVersionedEvents> buildVersionedEvents(

IngesterTwitterMessage twitterMessage) {

IngesterThriftVersionedEvents ingesterEvents =

new IngesterThriftVersionedEvents(twitterMessage.getUserId());

ingesterEvents.setDarkWrite(false);

ingesterEvents.setId(twitterMessage.getTweetId());

// We will emit both the original TwitterMessage, and the ThriftVersionedEvents instance, so we

// need to make sure they have separate DebugEvents copies.

ingesterEvents.setDebugEvents(twitterMessage.getDebugEvents().deepCopy());

try {

ThriftVersionedEvents versionedEvents =

messageConverter.convertMessageToThrift(twitterMessage, true, penguinVersionList);

ingesterEvents.setVersionedEvents(versionedEvents.getVersionedEvents());

return Optional.of(ingesterEvents);

} catch (IOException e) {

LOG.error("Failed to convert tweet " + twitterMessage.getTweetId() + " from TwitterMessage "

+ "to ThriftVersionedEvents for Penguin versions " + penguinVersionList,

e);

twitterMessageToTveErrorCount.increment();

}

return Optional.empty();

}

public void setThriftVersionedEventsBranchName(String thriftVersionedEventsBranchName) {

this.thriftVersionedEventsBranchName = thriftVersionedEventsBranchName;

}

}