package com.twitter.search.earlybird.archive;

import java.io.IOException;

import java.util.List;

import com.google.common.base.Preconditions;

import com.google.common.collect.Lists;

import com.twitter.search.common.partitioning.base.Segment;

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

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

import com.twitter.search.earlybird.EarlybirdIndexConfig;

import com.twitter.search.earlybird.archive.ArchiveTimeSlicer.ArchiveTimeSlice;

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

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

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

import com.twitter.search.earlybird.partition.DynamicPartitionConfig;

import com.twitter.search.earlybird.partition.SegmentInfo;

import com.twitter.search.earlybird.segment.EmptySegmentDataReaderSet;

import com.twitter.search.earlybird.segment.SegmentDataProvider;

import com.twitter.search.earlybird.segment.SegmentDataReaderSet;

public class ArchiveSegmentDataProvider implements SegmentDataProvider {

private static final org.slf4j.Logger LOG =

org.slf4j.LoggerFactory.getLogger(ArchiveSegmentDataProvider.class);

private DynamicPartitionConfig dynamicPartitionConfig;

private final ArchiveTimeSlicer timeSlicer;

private final DocumentFactory<ThriftIndexingEvent> documentFactory;

private final SegmentDataReaderSet readerSet;

public ArchiveSegmentDataProvider(

DynamicPartitionConfig dynamicPartitionConfig,

ArchiveTimeSlicer timeSlicer,

EarlybirdIndexConfig earlybirdIndexConfig) throws IOException {

this.dynamicPartitionConfig = dynamicPartitionConfig;

this.timeSlicer = timeSlicer;

this.readerSet = createSegmentDataReaderSet();

this.documentFactory = earlybirdIndexConfig.createDocumentFactory();

}

@Override

public List<Segment> newSegmentList() throws IOException {

List<ArchiveTimeSlice> timeSlices = timeSlicer.getTimeSlicesInTierRange();

if (timeSlices == null || timeSlices.isEmpty()) {

return Lists.newArrayList();

}

List<Segment> segments = Lists.newArrayListWithCapacity(timeSlices.size());

for (ArchiveTimeSlice timeSlice : timeSlices) {

segments.add(newArchiveSegment(timeSlice));

}

return segments;

}

/\*\*

\* Creates a new Segment instance for the given timeslice.

\*/

public ArchiveSegment newArchiveSegment(ArchiveTimeSlice archiveTimeSlice) {

return new ArchiveSegment(

archiveTimeSlice,

dynamicPartitionConfig.getCurrentPartitionConfig().getIndexingHashPartitionID(),

EarlybirdConfig.getMaxSegmentSize());

}

@Override

public SegmentDataReaderSet getSegmentDataReaderSet() {

return readerSet;

}

private EmptySegmentDataReaderSet createSegmentDataReaderSet() throws IOException {

return new EmptySegmentDataReaderSet() {

@Override

public RecordReader<TweetDocument> newDocumentReader(SegmentInfo segmentInfo)

throws IOException {

Segment segment = segmentInfo.getSegment();

Preconditions.checkArgument(segment instanceof ArchiveSegment);

return ((ArchiveSegment) segment).getStatusRecordReader(documentFactory);

}

};

}

}