package com.twitter.search.earlybird;

import java.io.IOException;

import com.google.common.base.Preconditions;

import org.apache.lucene.index.IndexWriterConfig;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.store.Directory;

import org.apache.lucene.store.RAMDirectory;

import com.twitter.decider.Decider;

import com.twitter.search.common.schema.DynamicSchema;

import com.twitter.search.common.schema.SearchWhitespaceAnalyzer;

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

import com.twitter.search.common.util.CloseResourceUtil;

import com.twitter.search.common.util.io.flushable.DataDeserializer;

import com.twitter.search.common.util.io.flushable.FlushInfo;

import com.twitter.search.core.earlybird.index.EarlybirdIndexSegmentData;

import com.twitter.search.core.earlybird.index.EarlybirdRealtimeIndexSegmentData;

import com.twitter.search.core.earlybird.index.extensions.EarlybirdIndexExtensionsFactory;

import com.twitter.search.core.earlybird.index.inverted.IndexOptimizer;

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

import com.twitter.search.earlybird.index.OptimizedTimeMapper;

import com.twitter.search.earlybird.index.OptimizedTweetIDMapper;

import com.twitter.search.earlybird.index.OutOfOrderRealtimeTweetIDMapper;

import com.twitter.search.earlybird.index.RealtimeTimeMapper;

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

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

/\*\*

\* Index config for the Real-Time in-memory Tweet cluster.

\*/

public class RealtimeEarlybirdIndexConfig extends EarlybirdIndexConfig {

private final CloseResourceUtil resourceCloser = new CloseResourceUtil();

public RealtimeEarlybirdIndexConfig(

EarlybirdCluster cluster, Decider decider, SearchIndexingMetricSet searchIndexingMetricSet,

CriticalExceptionHandler criticalExceptionHandler) {

super(cluster, decider, searchIndexingMetricSet, criticalExceptionHandler);

}

public RealtimeEarlybirdIndexConfig(

EarlybirdCluster cluster, DynamicSchema schema, Decider decider,

SearchIndexingMetricSet searchIndexingMetricSet,

CriticalExceptionHandler criticalExceptionHandler) {

super(cluster, schema, decider, searchIndexingMetricSet, criticalExceptionHandler);

}

@Override

public Directory newLuceneDirectory(SegmentSyncInfo segmentSyncInfo) {

return new RAMDirectory();

}

@Override

public IndexWriterConfig newIndexWriterConfig() {

return new IndexWriterConfig(new SearchWhitespaceAnalyzer())

.setSimilarity(IndexSearcher.getDefaultSimilarity());

}

@Override

public EarlybirdIndexSegmentData newSegmentData(

int maxSegmentSize,

long timeSliceID,

Directory dir,

EarlybirdIndexExtensionsFactory extensionsFactory) {

return new EarlybirdRealtimeIndexSegmentData(

maxSegmentSize,

timeSliceID,

getSchema(),

new OutOfOrderRealtimeTweetIDMapper(maxSegmentSize, timeSliceID),

new RealtimeTimeMapper(maxSegmentSize),

extensionsFactory);

}

@Override

public EarlybirdIndexSegmentData loadSegmentData(

FlushInfo flushInfo,

DataDeserializer dataInputStream,

Directory dir,

EarlybirdIndexExtensionsFactory extensionsFactory) throws IOException {

EarlybirdRealtimeIndexSegmentData.InMemorySegmentDataFlushHandler flushHandler;

boolean isOptimized = flushInfo.getBooleanProperty(

EarlybirdIndexSegmentData.AbstractSegmentDataFlushHandler.IS\_OPTIMIZED\_PROP\_NAME);

if (isOptimized) {

flushHandler = new EarlybirdRealtimeIndexSegmentData.InMemorySegmentDataFlushHandler(

getSchema(),

extensionsFactory,

new OptimizedTweetIDMapper.FlushHandler(),

new OptimizedTimeMapper.FlushHandler());

} else {

flushHandler = new EarlybirdRealtimeIndexSegmentData.InMemorySegmentDataFlushHandler(

getSchema(),

extensionsFactory,

new OutOfOrderRealtimeTweetIDMapper.FlushHandler(),

new RealtimeTimeMapper.FlushHandler());

}

return flushHandler.load(flushInfo, dataInputStream);

}

@Override

public EarlybirdIndexSegmentData optimize(

EarlybirdIndexSegmentData earlybirdIndexSegmentData) throws IOException {

Preconditions.checkArgument(

earlybirdIndexSegmentData instanceof EarlybirdRealtimeIndexSegmentData,

"Expected EarlybirdRealtimeIndexSegmentData but got %s",

earlybirdIndexSegmentData.getClass());

return IndexOptimizer.optimize((EarlybirdRealtimeIndexSegmentData) earlybirdIndexSegmentData);

}

@Override

public boolean isIndexStoredOnDisk() {

return false;

}

@Override

public final CloseResourceUtil getResourceCloser() {

return resourceCloser;

}

@Override

public boolean supportOutOfOrderIndexing() {

return true;

}

}