package com.twitter.search.core.earlybird.index;

import java.io.IOException;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import com.google.common.collect.Maps;

import org.apache.lucene.index.IndexWriterConfig;

import org.apache.lucene.search.IndexSearcher;

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

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

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

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

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

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

import com.twitter.search.core.earlybird.facets.AbstractFacetCountingArray;

import com.twitter.search.core.earlybird.facets.EarlybirdFacetDocValueSet;

import com.twitter.search.core.earlybird.facets.FacetCountingArray;

import com.twitter.search.core.earlybird.facets.FacetIDMap;

import com.twitter.search.core.earlybird.facets.FacetLabelProvider;

import com.twitter.search.core.earlybird.facets.FacetUtil;

import com.twitter.search.core.earlybird.facets.OptimizedFacetCountingArray;

import com.twitter.search.core.earlybird.index.column.DocValuesManager;

import com.twitter.search.core.earlybird.index.column.OptimizedDocValuesManager;

import com.twitter.search.core.earlybird.index.column.UnoptimizedDocValuesManager;

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

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

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

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

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

/\*\*

\* Implements {@link EarlybirdIndexSegmentData} for real-time in-memory Earlybird segments.

\*/

public class EarlybirdRealtimeIndexSegmentData extends EarlybirdIndexSegmentData {

private final EarlybirdRealtimeIndexExtensionsData indexExtension;

private EarlybirdFacetDocValueSet facetDocValueSet;

/\*\*

\* Creates a new empty real-time SegmentData instance.

\*/

public EarlybirdRealtimeIndexSegmentData(

int maxSegmentSize,

long timeSliceID,

Schema schema,

DocIDToTweetIDMapper docIdToTweetIdMapper,

TimeMapper timeMapper,

EarlybirdIndexExtensionsFactory indexExtensionsFactory) {

this(

maxSegmentSize,

timeSliceID,

schema,

false, // isOptimized

Integer.MAX\_VALUE,

new ConcurrentHashMap<>(),

new FacetCountingArray(maxSegmentSize),

new UnoptimizedDocValuesManager(schema, maxSegmentSize),

Maps.newHashMapWithExpectedSize(schema.getNumFacetFields()),

FacetIDMap.build(schema),

new DeletedDocs.Default(maxSegmentSize),

docIdToTweetIdMapper,

timeMapper,

indexExtensionsFactory == null

? null

: indexExtensionsFactory.newRealtimeIndexExtensionsData());

}

/\*\*

\* Creates a new real-time SegmentData instance using the passed in data structures. Usually this

\* constructor is used by the FlushHandler after a segment was loaded from disk, but also the

\* {@link IndexOptimizer} uses it to create an

\* optimized segment.

\*/

public EarlybirdRealtimeIndexSegmentData(

int maxSegmentSize,

long timeSliceID,

Schema schema,

boolean isOptimized,

int smallestDocID,

ConcurrentHashMap<String, InvertedIndex> perFieldMap,

AbstractFacetCountingArray facetCountingArray,

DocValuesManager docValuesManager,

Map<String, FacetLabelProvider> facetLabelProviders,

FacetIDMap facetIDMap,

DeletedDocs deletedDocs,

DocIDToTweetIDMapper docIdToTweetIdMapper,

TimeMapper timeMapper,

EarlybirdRealtimeIndexExtensionsData indexExtension) {

super(maxSegmentSize,

timeSliceID,

schema,

isOptimized,

smallestDocID,

perFieldMap,

new ConcurrentHashMap<>(),

facetCountingArray,

docValuesManager,

facetLabelProviders,

facetIDMap,

deletedDocs,

docIdToTweetIdMapper,

timeMapper);

this.indexExtension = indexExtension;

this.facetDocValueSet = null;

}

@Override

public EarlybirdRealtimeIndexExtensionsData getIndexExtensionsData() {

return indexExtension;

}

/\*\*

\* For realtime segments, this wraps a facet datastructure into a SortedSetDocValues to

\* comply to Lucene facet api.

\*/

public EarlybirdFacetDocValueSet getFacetDocValueSet() {

if (facetDocValueSet == null) {

AbstractFacetCountingArray facetCountingArray = getFacetCountingArray();

if (facetCountingArray != null) {

facetDocValueSet = new EarlybirdFacetDocValueSet(

facetCountingArray, getFacetLabelProviders(), getFacetIDMap());

}

}

return facetDocValueSet;

}

@Override

protected EarlybirdIndexSegmentAtomicReader doCreateAtomicReader() {

return new EarlybirdRealtimeIndexSegmentAtomicReader(this);

}

/\*\*

\* Convenience method for creating an EarlybirdIndexSegmentWriter for this segment with a default

\* IndexSegmentWriter config.

\*/

public EarlybirdIndexSegmentWriter createEarlybirdIndexSegmentWriter() {

return createEarlybirdIndexSegmentWriter(

new IndexWriterConfig(new SearchWhitespaceAnalyzer()).setSimilarity(

IndexSearcher.getDefaultSimilarity()));

}

@Override

public EarlybirdIndexSegmentWriter createEarlybirdIndexSegmentWriter(

IndexWriterConfig indexWriterConfig) {

// Prepare the in-memory segment with all enabled CSF fields.

DocValuesManager docValuesManager = getDocValuesManager();

for (Schema.FieldInfo fieldInfo : getSchema().getFieldInfos()) {

if (fieldInfo.getFieldType().getCsfType() != null) {

docValuesManager.addColumnStrideField(fieldInfo.getName(), fieldInfo.getFieldType());

}

}

return new EarlybirdRealtimeIndexSegmentWriter(

this,

indexWriterConfig.getAnalyzer(),

indexWriterConfig.getSimilarity());

}

@Override

public EarlybirdIndexSegmentData.AbstractSegmentDataFlushHandler getFlushHandler() {

return new InMemorySegmentDataFlushHandler(this);

}

public static class InMemorySegmentDataFlushHandler

extends AbstractSegmentDataFlushHandler<EarlybirdRealtimeIndexExtensionsData> {

public InMemorySegmentDataFlushHandler(EarlybirdIndexSegmentData objectToFlush) {

super(objectToFlush);

}

public InMemorySegmentDataFlushHandler(

Schema schema,

EarlybirdIndexExtensionsFactory factory,

Flushable.Handler<? extends DocIDToTweetIDMapper> docIdMapperFlushHandler,

Flushable.Handler<? extends TimeMapper> timeMapperFlushHandler) {

super(schema, factory, docIdMapperFlushHandler, timeMapperFlushHandler);

}

@Override

protected EarlybirdRealtimeIndexExtensionsData newIndexExtension() {

return indexExtensionsFactory.newRealtimeIndexExtensionsData();

}

@Override

protected void flushAdditionalDataStructures(

FlushInfo flushInfo,

DataSerializer out,

EarlybirdIndexSegmentData segmentData) throws IOException {

segmentData.getFacetCountingArray().getFlushHandler()

.flush(flushInfo.newSubProperties("facet\_counting\_array"), out);

// flush all column stride fields

segmentData.getDocValuesManager().getFlushHandler()

.flush(flushInfo.newSubProperties("doc\_values"), out);

segmentData.getFacetIDMap().getFlushHandler()

.flush(flushInfo.newSubProperties("facet\_id\_map"), out);

segmentData.getDeletedDocs().getFlushHandler()

.flush(flushInfo.newSubProperties("deleted\_docs"), out);

}

@Override

protected EarlybirdIndexSegmentData constructSegmentData(

FlushInfo flushInfo,

ConcurrentHashMap<String, InvertedIndex> perFieldMap,

int maxSegmentSize,

EarlybirdRealtimeIndexExtensionsData indexExtension,

DocIDToTweetIDMapper docIdToTweetIdMapper,

TimeMapper timeMapper,

DataDeserializer in) throws IOException {

boolean isOptimized = flushInfo.getBooleanProperty(IS\_OPTIMIZED\_PROP\_NAME);

Flushable.Handler<? extends AbstractFacetCountingArray> facetLoader = isOptimized

? new OptimizedFacetCountingArray.FlushHandler()

: new FacetCountingArray.FlushHandler(maxSegmentSize);

AbstractFacetCountingArray facetCountingArray =

facetLoader.load(flushInfo.getSubProperties("facet\_counting\_array"), in);

Flushable.Handler<? extends DocValuesManager> docValuesLoader = isOptimized

? new OptimizedDocValuesManager.OptimizedFlushHandler(schema)

: new UnoptimizedDocValuesManager.UnoptimizedFlushHandler(schema);

DocValuesManager docValuesManager =

docValuesLoader.load(flushInfo.getSubProperties("doc\_values"), in);

FacetIDMap facetIDMap = new FacetIDMap.FlushHandler(schema)

.load(flushInfo.getSubProperties("facet\_id\_map"), in);

DeletedDocs.Default deletedDocs = new DeletedDocs.Default.FlushHandler(maxSegmentSize)

.load(flushInfo.getSubProperties("deleted\_docs"), in);

return new EarlybirdRealtimeIndexSegmentData(

maxSegmentSize,

flushInfo.getLongProperty(TIME\_SLICE\_ID\_PROP\_NAME),

schema,

isOptimized,

flushInfo.getIntProperty(SMALLEST\_DOCID\_PROP\_NAME),

perFieldMap,

facetCountingArray,

docValuesManager,

FacetUtil.getFacetLabelProviders(schema, perFieldMap),

facetIDMap,

deletedDocs,

docIdToTweetIdMapper,

timeMapper,

indexExtension);

}

}

}