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

import java.io.IOException;

import java.util.concurrent.ConcurrentHashMap;

import org.apache.lucene.index.IndexWriterConfig;

import org.apache.lucene.index.LeafReader;

import org.apache.lucene.store.Directory;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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.FacetCountingArrayWriter;

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

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.extensions.EarlybirdIndexExtensionsData;

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

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

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

/\*\*

\* Implements {@link EarlybirdIndexSegmentData} for Lucene-based on-disk Earlybird segments.

\*/

public final class EarlybirdLuceneIndexSegmentData extends EarlybirdIndexSegmentData {

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

private final Directory directory;

private final EarlybirdIndexExtensionsData indexExtension;

/\*\*

\* Creates a new Lucene-based SegmentData instance from a lucene directory.

\*/

public EarlybirdLuceneIndexSegmentData(

Directory directory,

int maxSegmentSize,

long timeSliceID,

Schema schema,

DocIDToTweetIDMapper docIdToTweetIdMapper,

TimeMapper timeMapper,

EarlybirdIndexExtensionsFactory indexExtensionsFactory) {

this(

directory,

maxSegmentSize,

timeSliceID,

schema,

false, // isOptimized

0, // smallestDocId

new ConcurrentHashMap<>(),

AbstractFacetCountingArray.EMPTY\_ARRAY,

new OptimizedDocValuesManager(schema, maxSegmentSize),

docIdToTweetIdMapper,

timeMapper,

indexExtensionsFactory == null

? null : indexExtensionsFactory.newLuceneIndexExtensionsData());

}

public EarlybirdLuceneIndexSegmentData(

Directory directory,

int maxSegmentSize,

long timeSliceID,

Schema schema,

boolean isOptimized,

int smallestDocID,

ConcurrentHashMap<String, InvertedIndex> perFieldMap,

AbstractFacetCountingArray facetCountingArray,

DocValuesManager docValuesManager,

DocIDToTweetIDMapper docIdToTweetIdMapper,

TimeMapper timeMapper,

EarlybirdIndexExtensionsData indexExtension) {

super(maxSegmentSize,

timeSliceID,

schema,

isOptimized,

smallestDocID,

perFieldMap,

new ConcurrentHashMap<>(),

facetCountingArray,

docValuesManager,

null, // facetLabelProviders

null, // facetIDMap

DeletedDocs.NO\_DELETES,

docIdToTweetIdMapper,

timeMapper);

this.directory = directory;

this.indexExtension = indexExtension;

}

public Directory getLuceneDirectory() {

return directory;

}

@Override

public EarlybirdIndexExtensionsData getIndexExtensionsData() {

return indexExtension;

}

@Override

public FacetCountingArrayWriter createFacetCountingArrayWriter() {

return null;

}

@Override

protected EarlybirdIndexSegmentAtomicReader doCreateAtomicReader() throws IOException {

// EarlybirdSegment creates one single EarlybirdIndexSegmentAtomicReader instance per segment

// and caches it, and the cached instance is recreated only when the segment's data changes.

// This is why this is a good place to reload all CSFs that should be loaded in RAM. Also, it's

// easier and less error-prone to do it here, than trying to track down all places that mutate

// the segment data and do it there.

LeafReader reader = getLeafReaderFromOptimizedDirectory(directory);

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

// Load CSF into RAM based on configurations in the schema.

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

&& fieldInfo.getFieldType().isCsfLoadIntoRam()) {

if (reader.getNumericDocValues(fieldInfo.getName()) != null) {

ColumnStrideFieldIndex index = getDocValuesManager().addColumnStrideField(

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

index.load(reader, fieldInfo.getName());

} else {

LOG.warn("Field {} does not have NumericDocValues.", fieldInfo.getName());

}

}

}

return new EarlybirdLuceneIndexSegmentAtomicReader(this, directory);

}

@Override

public EarlybirdIndexSegmentWriter createEarlybirdIndexSegmentWriter(

IndexWriterConfig indexWriterConfig) throws IOException {

return new EarlybirdLuceneIndexSegmentWriter(this, indexWriterConfig);

}

@Override

public EarlybirdIndexSegmentData.AbstractSegmentDataFlushHandler getFlushHandler() {

return new OnDiskSegmentDataFlushHandler(this);

}

public static class OnDiskSegmentDataFlushHandler

extends AbstractSegmentDataFlushHandler<EarlybirdIndexExtensionsData> {

private final Directory directory;

public OnDiskSegmentDataFlushHandler(EarlybirdLuceneIndexSegmentData objectToFlush) {

super(objectToFlush);

this.directory = objectToFlush.directory;

}

public OnDiskSegmentDataFlushHandler(

Schema schema,

Directory directory,

EarlybirdIndexExtensionsFactory indexExtensionsFactory,

Flushable.Handler<? extends DocIDToTweetIDMapper> docIdMapperFlushHandler,

Flushable.Handler<? extends TimeMapper> timeMapperFlushHandler) {

super(schema, indexExtensionsFactory, docIdMapperFlushHandler, timeMapperFlushHandler);

this.directory = directory;

}

@Override

protected EarlybirdIndexExtensionsData newIndexExtension() {

return indexExtensionsFactory.newLuceneIndexExtensionsData();

}

@Override

protected void flushAdditionalDataStructures(

FlushInfo flushInfo, DataSerializer out, EarlybirdIndexSegmentData toFlush) {

}

@Override

protected EarlybirdIndexSegmentData constructSegmentData(

FlushInfo flushInfo,

ConcurrentHashMap<String, InvertedIndex> perFieldMap,

int maxSegmentSize,

EarlybirdIndexExtensionsData indexExtension,

DocIDToTweetIDMapper docIdToTweetIdMapper,

TimeMapper timeMapper,

DataDeserializer in) {

return new EarlybirdLuceneIndexSegmentData(

directory,

maxSegmentSize,

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

schema,

flushInfo.getBooleanProperty(IS\_OPTIMIZED\_PROP\_NAME),

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

perFieldMap,

AbstractFacetCountingArray.EMPTY\_ARRAY,

new OptimizedDocValuesManager(schema, maxSegmentSize),

docIdToTweetIdMapper,

timeMapper,

indexExtension);

}

}

}