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

import java.io.IOException;

import com.google.common.base.Preconditions;

import org.apache.lucene.index.BinaryDocValues;

import org.apache.lucene.index.FieldInfos;

import org.apache.lucene.index.FilterLeafReader;

import org.apache.lucene.index.LeafMetaData;

import org.apache.lucene.index.LeafReader;

import org.apache.lucene.index.NumericDocValues;

import org.apache.lucene.index.PointValues;

import org.apache.lucene.index.PostingsEnum;

import org.apache.lucene.index.SortedDocValues;

import org.apache.lucene.index.SortedNumericDocValues;

import org.apache.lucene.index.SortedSetDocValues;

import org.apache.lucene.index.StoredFieldVisitor;

import org.apache.lucene.index.Term;

import org.apache.lucene.index.Terms;

import org.apache.lucene.index.TermsEnum;

import org.apache.lucene.search.DocIdSetIterator;

import org.apache.lucene.store.Directory;

import org.apache.lucene.util.Bits;

import org.apache.lucene.util.BytesRef;

import com.twitter.search.common.encoding.docvalues.CSFTypeUtil;

import com.twitter.search.common.encoding.features.IntegerEncodedFeatures;

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

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

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

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

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

public final class EarlybirdLuceneIndexSegmentAtomicReader

extends EarlybirdIndexSegmentAtomicReader {

private abstract static class DocIdSetIteratorWrapper extends NumericDocValues {

private final DocIdSetIterator delegate;

public DocIdSetIteratorWrapper(DocIdSetIterator delegate) {

this.delegate = Preconditions.checkNotNull(delegate);

}

@Override

public int docID() {

return delegate.docID();

}

@Override

public int nextDoc() throws IOException {

return delegate.nextDoc();

}

@Override

public int advance(int target) throws IOException {

return delegate.advance(target);

}

@Override

public long cost() {

return delegate.cost();

}

}

private static class BytesRefBasedIntegerEncodedFeatures extends IntegerEncodedFeatures {

private final BytesRef bytesRef;

private final int numInts;

public BytesRefBasedIntegerEncodedFeatures(BytesRef bytesRef, int numInts) {

this.bytesRef = bytesRef;

this.numInts = numInts;

}

@Override

public int getInt(int pos) {

return CSFTypeUtil.convertFromBytes(bytesRef.bytes, bytesRef.offset, pos);

}

@Override

public void setInt(int pos, int value) {

throw new UnsupportedOperationException();

}

@Override

public int getNumInts() {

return numInts;

}

}

private static final int OLDEST\_DOC\_SKIP\_INTERVAL = 256;

private final LeafReader delegate;

/\*\*

\* Do not add public constructors to this class. EarlybirdLuceneIndexSegmentAtomicReader instances

\* should be created only by calling EarlybirdLuceneIndexSegmentData.createAtomicReader(), to make

\* sure everything is set up properly (such as CSF readers).

\*/

EarlybirdLuceneIndexSegmentAtomicReader(

EarlybirdIndexSegmentData segmentData, Directory directory) throws IOException {

super(segmentData);

this.delegate = getDelegateReader(directory);

}

private LeafReader getDelegateReader(Directory directory) throws IOException {

LeafReader directoryReader =

EarlybirdIndexSegmentData.getLeafReaderFromOptimizedDirectory(directory);

return new FilterLeafReader(directoryReader) {

@Override

public NumericDocValues getNumericDocValues(String field) throws IOException {

EarlybirdFieldType type = getSchema().getFieldInfo(field).getFieldType();

if ((type == null) || !type.isCsfViewField()) {

return in.getNumericDocValues(field);

}

// Compute as many things as possible once, outside the NumericDocValues.get() call.

String baseFieldName = getSchema().getFieldInfo(type.getCsfViewBaseFieldId()).getName();

FieldInfo baseFieldInfo =

Preconditions.checkNotNull(getSchema().getFieldInfo(baseFieldName));

EarlybirdFieldType baseFieldType = baseFieldInfo.getFieldType();

Preconditions.checkState(!baseFieldType.isCsfVariableLength());

int numInts = baseFieldType.getCsfFixedLengthNumValuesPerDoc();

FeatureConfiguration featureConfiguration =

Preconditions.checkNotNull(type.getCsfViewFeatureConfiguration());

Preconditions.checkArgument(featureConfiguration.getValueIndex() < numInts);

if (numInts == 1) {

// All encoded tweet features are encoded in a single integer.

NumericDocValues numericDocValues = in.getNumericDocValues(baseFieldName);

return new DocIdSetIteratorWrapper(numericDocValues) {

@Override

public long longValue() throws IOException {

return (numericDocValues.longValue() & featureConfiguration.getBitMask())

>> featureConfiguration.getBitStartPosition();

}

@Override

public boolean advanceExact(int target) throws IOException {

return numericDocValues.advanceExact(target);

}

};

}

BinaryDocValues binaryDocValues =

Preconditions.checkNotNull(in.getBinaryDocValues(baseFieldName));

return new DocIdSetIteratorWrapper(binaryDocValues) {

@Override

public long longValue() throws IOException {

BytesRef data = binaryDocValues.binaryValue();

IntegerEncodedFeatures encodedFeatures =

new BytesRefBasedIntegerEncodedFeatures(data, numInts);

return encodedFeatures.getFeatureValue(featureConfiguration);

}

@Override

public boolean advanceExact(int target) throws IOException {

return binaryDocValues.advanceExact(target);

}

};

}

@Override

public CacheHelper getCoreCacheHelper() {

return in.getCoreCacheHelper();

}

@Override

public CacheHelper getReaderCacheHelper() {

return in.getReaderCacheHelper();

}

};

}

private TermsEnum getTermsEnumAtTerm(Term term) throws IOException {

Terms terms = terms(term.field());

if (terms == null) {

return null;

}

TermsEnum termsEnum = terms.iterator();

return termsEnum.seekExact(term.bytes()) ? termsEnum : null;

}

@Override

public int getOldestDocID(Term term) throws IOException {

TermsEnum termsEnum = getTermsEnumAtTerm(term);

if (termsEnum == null) {

return EarlybirdIndexSegmentAtomicReader.TERM\_NOT\_FOUND;

}

PostingsEnum td = termsEnum.postings(null);

int oldestDocID = td.nextDoc();

if (oldestDocID == DocIdSetIterator.NO\_MORE\_DOCS) {

return EarlybirdIndexSegmentAtomicReader.TERM\_NOT\_FOUND;

}

final int docFreq = termsEnum.docFreq();

if (docFreq > OLDEST\_DOC\_SKIP\_INTERVAL \* 16) {

final int skipSize = docFreq / OLDEST\_DOC\_SKIP\_INTERVAL;

do {

oldestDocID = td.docID();

} while (td.advance(oldestDocID + skipSize) != DocIdSetIterator.NO\_MORE\_DOCS);

td = delegate.postings(term);

td.advance(oldestDocID);

}

do {

oldestDocID = td.docID();

} while (td.nextDoc() != DocIdSetIterator.NO\_MORE\_DOCS);

return oldestDocID;

}

@Override

public int getTermID(Term term) throws IOException {

TermsEnum termsEnum = getTermsEnumAtTerm(term);

return termsEnum != null

? (int) termsEnum.ord()

: EarlybirdIndexSegmentAtomicReader.TERM\_NOT\_FOUND;

}

@Override

public Terms terms(String field) throws IOException {

return delegate.terms(field);

}

@Override

public FieldInfos getFieldInfos() {

return delegate.getFieldInfos();

}

@Override

public Bits getLiveDocs() {

return getDeletesView().getLiveDocs();

}

@Override

public int numDocs() {

return delegate.numDocs();

}

@Override

public int maxDoc() {

return delegate.maxDoc();

}

@Override

public void document(int docID, StoredFieldVisitor visitor) throws IOException {

delegate.document(docID, visitor);

}

@Override

public boolean hasDeletions() {

return getDeletesView().hasDeletions();

}

@Override

protected void doClose() throws IOException {

delegate.close();

}

@Override

public NumericDocValues getNumericDocValues(String field) throws IOException {

FieldInfo fieldInfo = getSegmentData().getSchema().getFieldInfo(field);

if (fieldInfo == null) {

return null;

}

// If this field is a CSF view field or if it's not loaded in memory, get the NumericDocValues

// from the delegate.

EarlybirdFieldType fieldType = fieldInfo.getFieldType();

if (fieldType.isCsfViewField() || !fieldInfo.getFieldType().isCsfLoadIntoRam()) {

NumericDocValues delegateVals = delegate.getNumericDocValues(field);

if (delegateVals != null) {

return delegateVals;

}

}

// The field is either loaded in memory, or the delegate doesn't have NumericDocValues for it.

// Return the NumericDocValues for this field stored in the DocValuesManager.

ColumnStrideFieldIndex csf =

getSegmentData().getDocValuesManager().getColumnStrideFieldIndex(field);

return csf != null ? new ColumnStrideFieldDocValues(csf, this) : null;

}

@Override

public BinaryDocValues getBinaryDocValues(String field) throws IOException {

return delegate.getBinaryDocValues(field);

}

@Override

public SortedDocValues getSortedDocValues(String field) throws IOException {

return delegate.getSortedDocValues(field);

}

@Override

public SortedSetDocValues getSortedSetDocValues(String field) throws IOException {

return delegate.getSortedSetDocValues(field);

}

@Override

public NumericDocValues getNormValues(String field) throws IOException {

return delegate.getNormValues(field);

}

@Override

public SortedNumericDocValues getSortedNumericDocValues(String field) throws IOException {

return delegate.getSortedNumericDocValues(field);

}

@Override

public void checkIntegrity() throws IOException {

delegate.checkIntegrity();

}

@Override

public PointValues getPointValues(String field) throws IOException {

return delegate.getPointValues(field);

}

@Override

public LeafMetaData getMetaData() {

return delegate.getMetaData();

}

@Override

public CacheHelper getCoreCacheHelper() {

return delegate.getCoreCacheHelper();

}

@Override

public CacheHelper getReaderCacheHelper() {

return delegate.getReaderCacheHelper();

}

}