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

import org.apache.lucene.search.DocIdSetIterator;

/\*\*

\* A skip list reader of a single term used {@link HighDFPackedIntsDocsEnum}.

\* @see HighDFPackedIntsPostingLists

\*/

class HighDFPackedIntsSkipListReader {

/\*\* Skip lists int pool. \*/

private final IntBlockPool skipLists;

/\*\* Whether positions are omitted in the posting list having the read skip list. \*/

private final boolean omitPositions;

/\*\*

\* Last doc in the previous slice relative to the current delta-freq slice. This value is 0 if

\* the current slice is the first delta-freq slice.

\*/

private int previousDocIDCurrentSlice;

/\*\* Encoded metadata of the current delta-freq slice.\*/

private int encodedMetadataCurrentSlice;

/\*\*

\* Pointer to the first int (contains the position slice header) of the position slice that has

\* the first position of the first doc in the current delta-freq slice.

\*/

private int positionCurrentSliceIndex;

/\*\* Pointer to the first int in the current delta-freq slice. \*/

private int deltaFreqCurrentSlicePointer;

/\*\* Data of next slice. \*/

private int previousDocIDNextSlice;

private int encodedMetadataNextSlice;

private int positionNextSliceIndex;

private int deltaFreqNextSlicePointer;

/\*\* Used to load blocks and read ints from skip lists int pool. \*/

private int[] currentSkipListBlock;

private int skipListBlockStart;

private int skipListBlockIndex;

/\*\* Number of remaining skip entries for the read skip list. \*/

private int numSkipListEntriesRemaining;

/\*\* Largest doc ID in the posting list having the read skip list. \*/

private final int largestDocID;

/\*\* Pointer to the first int in the first slice that stores positions for this term. \*/

private final int positionListPointer;

/\*\* Total number of docs in the posting list having the read skip list. \*/

private final int numDocsTotal;

/\*\*

\* Create a skip list reader specified by the given skip list pointer in the given skip lists int

\* pool.

\*

\* @param skipLists int pool where the read skip list exists

\* @param skipListPointer pointer to the read skip list

\* @param omitPositions whether positions are omitted in the positing list to which the read skip

\* list belongs

\*/

public HighDFPackedIntsSkipListReader(

final IntBlockPool skipLists,

final int skipListPointer,

final boolean omitPositions) {

this.skipLists = skipLists;

this.omitPositions = omitPositions;

this.skipListBlockStart = IntBlockPool.getBlockStart(skipListPointer);

this.skipListBlockIndex = IntBlockPool.getOffsetInBlock(skipListPointer);

this.currentSkipListBlock = skipLists.getBlock(skipListBlockStart);

// Read skip list header.

this.numSkipListEntriesRemaining = readNextValueFromSkipListBlock();

this.largestDocID = readNextValueFromSkipListBlock();

this.numDocsTotal = readNextValueFromSkipListBlock();

int deltaFreqListPointer = readNextValueFromSkipListBlock();

this.positionListPointer = omitPositions ? -1 : readNextValueFromSkipListBlock();

// Set it back by one slice for fetchNextSkipEntry() to advance correctly.

this.deltaFreqNextSlicePointer = deltaFreqListPointer - HighDFPackedIntsPostingLists.SLICE\_SIZE;

fetchNextSkipEntry();

}

/\*\*

\* Load already fetched data in next skip entry into current data variables, and pre-fetch again.

\*/

public void getNextSkipEntry() {

previousDocIDCurrentSlice = previousDocIDNextSlice;

encodedMetadataCurrentSlice = encodedMetadataNextSlice;

positionCurrentSliceIndex = positionNextSliceIndex;

deltaFreqCurrentSlicePointer = deltaFreqNextSlicePointer;

fetchNextSkipEntry();

}

/\*\*

\* Fetch data for next skip entry if skip list is not exhausted; otherwise, set docIDNextSlice

\* to NO\_MORE\_DOCS.

\*/

private void fetchNextSkipEntry() {

if (numSkipListEntriesRemaining == 0) {

previousDocIDNextSlice = DocIdSetIterator.NO\_MORE\_DOCS;

return;

}

previousDocIDNextSlice = readNextValueFromSkipListBlock();

encodedMetadataNextSlice = readNextValueFromSkipListBlock();

if (!omitPositions) {

positionNextSliceIndex = readNextValueFromSkipListBlock();

}

deltaFreqNextSlicePointer += HighDFPackedIntsPostingLists.SLICE\_SIZE;

numSkipListEntriesRemaining--;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Getters of data in skip list entry \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* In the context of a current slice, this is the docID of the last document in the previous

\* slice (or 0 if the current slice is the first slice).

\*

\* @see HighDFPackedIntsPostingLists#SKIPLIST\_ENTRY\_SIZE

\*/

public int getPreviousDocIDCurrentSlice() {

return previousDocIDCurrentSlice;

}

/\*\*

\* Get the encoded metadata of the current delta-freq slice.

\*

\* @see HighDFPackedIntsPostingLists#SKIPLIST\_ENTRY\_SIZE

\*/

public int getEncodedMetadataCurrentSlice() {

return encodedMetadataCurrentSlice;

}

/\*\*

\* Get the pointer to the first int, WHICH CONTAINS THE POSITION SLICE HEADER, of the position

\* slice that contains the first position of the first doc in the delta-freq slice that

\* is corresponding to the current skip list entry.

\*

\* @see HighDFPackedIntsPostingLists#SKIPLIST\_ENTRY\_SIZE

\*/

public int getPositionCurrentSlicePointer() {

assert !omitPositions;

return positionListPointer

+ positionCurrentSliceIndex \* HighDFPackedIntsPostingLists.SLICE\_SIZE;

}

/\*\*

\* Get the pointer to the first int in the current delta-freq slice.

\*/

public int getDeltaFreqCurrentSlicePointer() {

return deltaFreqCurrentSlicePointer;

}

/\*\*

\* In the context of next slice, get the last doc ID in the previous slice. This is used to skip

\* over slices.

\*

\* @see HighDFPackedIntsDocsEnum#skipTo(int)

\*/

public int peekPreviousDocIDNextSlice() {

return previousDocIDNextSlice;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Getters of data in skip list header \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public int getLargestDocID() {

return largestDocID;

}

public int getNumDocsTotal() {

return numDocsTotal;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Methods helping loading int block and read ints \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private int readNextValueFromSkipListBlock() {

if (skipListBlockIndex == IntBlockPool.BLOCK\_SIZE) {

loadSkipListBlock();

}

return currentSkipListBlock[skipListBlockIndex++];

}

private void loadSkipListBlock() {

skipListBlockStart += IntBlockPool.BLOCK\_SIZE;

currentSkipListBlock = skipLists.getBlock(skipListBlockStart);

skipListBlockIndex = 0;

}

}