package com.twitter.search.earlybird.querycache;

import java.io.IOException;

import org.apache.lucene.search.DocIdSetIterator;

public class CachedResultDocIdSetIterator extends DocIdSetIterator {

// With the realtime index, we grow the doc id negatively.

// Hence the smallest doc id is the ID the latest/newest document in the cache.

private final int cachedSmallestDocID;

// Documents that were indexed after the last cache update

private final DocIdSetIterator freshDocIdIterator;

// Documents that were cached

private final DocIdSetIterator cachedDocIdIterator;

private int currentDocId;

private boolean initialized = false;

public CachedResultDocIdSetIterator(int cachedSmallestDocID,

DocIdSetIterator freshDocIdIterator,

DocIdSetIterator cachedDocIdIterator) {

this.cachedSmallestDocID = cachedSmallestDocID;

this.freshDocIdIterator = freshDocIdIterator;

this.cachedDocIdIterator = cachedDocIdIterator;

this.currentDocId = -1;

}

@Override

public int docID() {

return currentDocId;

}

@Override

public int nextDoc() throws IOException {

if (currentDocId < cachedSmallestDocID) {

currentDocId = freshDocIdIterator.nextDoc();

} else if (currentDocId != NO\_MORE\_DOCS) {

if (!initialized) {

// the first time we come in here, currentDocId should be pointing to

// something >= cachedMinDocID. We need to go to the doc after cachedMinDocID.

currentDocId = cachedDocIdIterator.advance(currentDocId + 1);

initialized = true;

} else {

currentDocId = cachedDocIdIterator.nextDoc();

}

}

return currentDocId;

}

@Override

public int advance(int target) throws IOException {

if (target < cachedSmallestDocID) {

currentDocId = freshDocIdIterator.advance(target);

} else if (currentDocId != NO\_MORE\_DOCS) {

initialized = true;

currentDocId = cachedDocIdIterator.advance(target);

}

return currentDocId;

}

@Override

public long cost() {

if (currentDocId < cachedSmallestDocID) {

return freshDocIdIterator.cost();

} else {

return cachedDocIdIterator.cost();

}

}

}