package com.twitter.search.earlybird.search.queries;

import java.io.IOException;

import javax.annotation.Nullable;

import com.google.common.annotations.VisibleForTesting;

import org.apache.lucene.search.DocIdSetIterator;

import com.twitter.common.util.Clock;

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.common.search.EarlyTerminationState;

import com.twitter.search.common.search.TerminationTracker;

import com.twitter.search.earlybird.common.config.EarlybirdConfig;

/\*\*

\* DocIdSetIterator whose nextDoc() and advance() will early terminate by returning NO\_MORE\_DOCS

\* after the given deadline.

\*/

public class TimedDocIdSetIterator extends DocIdSetIterator {

// check deadline every NEXT\_CALL\_TIMEOUT\_CHECK\_PERIOD calls to nextDoc()

@VisibleForTesting

protected static final int NEXT\_CALL\_TIMEOUT\_CHECK\_PERIOD =

EarlybirdConfig.getInt("timed\_doc\_id\_set\_next\_doc\_deadline\_check\_period", 1000);

// check deadline every ADVANCE\_CALL\_TIMEOUT\_CHECK\_PERIOD calls to advance()

private static final int ADVANCE\_CALL\_TIMEOUT\_CHECK\_PERIOD =

EarlybirdConfig.getInt("timed\_doc\_id\_set\_advance\_deadline\_check\_period", 100);

private final Clock clock;

private final DocIdSetIterator innerIterator;

private final SearchCounter timeoutCountStat;

@Nullable

private final TerminationTracker terminationTracker;

private final long deadlineMillisFromEpoch;

private int docId = -1;

private int nextCounter = 0;

private int advanceCounter = 0;

public TimedDocIdSetIterator(DocIdSetIterator innerIterator,

@Nullable TerminationTracker terminationTracker,

final long timeoutOverride,

@Nullable SearchCounter timeoutCountStat) {

this(innerIterator, terminationTracker, timeoutOverride, timeoutCountStat, Clock.SYSTEM\_CLOCK);

}

protected TimedDocIdSetIterator(DocIdSetIterator innerIterator,

@Nullable TerminationTracker terminationTracker,

final long timeoutOverride,

@Nullable SearchCounter timeoutCountStat,

Clock clock) {

this.clock = clock;

this.innerIterator = innerIterator;

this.timeoutCountStat = timeoutCountStat;

this.terminationTracker = terminationTracker;

if (terminationTracker == null) {

deadlineMillisFromEpoch = -1;

} else {

if (timeoutOverride > 0) {

deadlineMillisFromEpoch = terminationTracker.getClientStartTimeMillis() + timeoutOverride;

} else {

deadlineMillisFromEpoch = terminationTracker.getTimeoutEndTimeWithReservation();

}

}

}

@VisibleForTesting

protected TimedDocIdSetIterator(DocIdSetIterator innerIterator,

final long deadline,

@Nullable SearchCounter timeoutCountStat,

Clock clock) {

this.clock = clock;

this.innerIterator = innerIterator;

this.timeoutCountStat = timeoutCountStat;

this.terminationTracker = null;

this.deadlineMillisFromEpoch = deadline;

}

@Override

public int docID() {

return docId;

}

@Override

public int nextDoc() throws IOException {

if (++nextCounter % NEXT\_CALL\_TIMEOUT\_CHECK\_PERIOD == 0

&& clock.nowMillis() > deadlineMillisFromEpoch) {

if (timeoutCountStat != null) {

timeoutCountStat.increment();

}

if (terminationTracker != null) {

terminationTracker.setEarlyTerminationState(

EarlyTerminationState.TERMINATED\_TIME\_OUT\_EXCEEDED);

}

return docId = NO\_MORE\_DOCS;

}

return docId = innerIterator.nextDoc();

}

@Override

public int advance(int target) throws IOException {

if (++advanceCounter % ADVANCE\_CALL\_TIMEOUT\_CHECK\_PERIOD == 0

&& clock.nowMillis() > deadlineMillisFromEpoch) {

if (timeoutCountStat != null) {

timeoutCountStat.increment();

}

if (terminationTracker != null) {

terminationTracker.setEarlyTerminationState(

EarlyTerminationState.TERMINATED\_TIME\_OUT\_EXCEEDED);

}

return docId = NO\_MORE\_DOCS;

}

return docId = innerIterator.advance(target);

}

@Override

public long cost() {

return innerIterator.cost();

}

}