package com.twitter.search.earlybird\_root.routers;

import java.util.concurrent.TimeUnit;

import javax.inject.Inject;

import javax.inject.Named;

import com.google.common.base.Preconditions;

import com.twitter.common.util.Clock;

import com.twitter.finagle.Service;

import com.twitter.search.common.decider.SearchDecider;

import com.twitter.search.common.partitioning.snowflakeparser.SnowflakeIdParser;

import com.twitter.search.common.query.thriftjava.CollectorTerminationParams;

import com.twitter.search.earlybird.thrift.EarlybirdRequest;

import com.twitter.search.earlybird.thrift.EarlybirdResponse;

import com.twitter.search.earlybird.thrift.ThriftSearchRankingMode;

import com.twitter.search.earlybird.thrift.ThriftSearchResult;

import com.twitter.search.earlybird\_root.common.EarlybirdFeatureSchemaMerger;

import com.twitter.search.earlybird\_root.common.EarlybirdRequestContext;

import com.twitter.search.earlybird\_root.common.InjectionNames;

import com.twitter.search.earlybird\_root.filters.EarlybirdTimeRangeFilter;

public class RelevanceRequestRouter extends AbstractRecencyAndRelevanceRequestRouter {

private static final long MILLIS\_IN\_ONE\_DAY = TimeUnit.DAYS.toMillis(1);

@Inject

public RelevanceRequestRouter(

@Named(InjectionNames.REALTIME)

Service<EarlybirdRequestContext, EarlybirdResponse> realtime,

@Named(InjectionNames.PROTECTED)

Service<EarlybirdRequestContext, EarlybirdResponse> protectedRealtime,

@Named(InjectionNames.FULL\_ARCHIVE)

Service<EarlybirdRequestContext, EarlybirdResponse> fullArchive,

@Named(RelevanceRequestRouterModule.REALTIME\_TIME\_RANGE\_FILTER)

EarlybirdTimeRangeFilter realtimeTimeRangeFilter,

@Named(RelevanceRequestRouterModule.PROTECTED\_TIME\_RANGE\_FILTER)

EarlybirdTimeRangeFilter protectedTimeRangeFilter,

@Named(RelevanceRequestRouterModule.FULL\_ARCHIVE\_TIME\_RANGE\_FILTER)

EarlybirdTimeRangeFilter fullArchiveTimeRangeFilter,

Clock clock,

SearchDecider decider,

EarlybirdFeatureSchemaMerger featureSchemaMerger) {

super(realtime,

protectedRealtime,

fullArchive,

realtimeTimeRangeFilter,

protectedTimeRangeFilter,

fullArchiveTimeRangeFilter,

ThriftSearchRankingMode.RELEVANCE,

clock,

decider,

featureSchemaMerger);

}

@Override

protected boolean shouldSendRequestToFullArchiveCluster(

EarlybirdRequest request, EarlybirdResponse realtimeResponse) {

int numResultsRequested = request.getSearchQuery().getNumResults();

int numHitsProcessed = realtimeResponse.getSearchResults().isSetNumHitsProcessed()

? realtimeResponse.getSearchResults().getNumHitsProcessed()

: -1;

if (numHitsProcessed < numResultsRequested) {

// Send query to the full archive cluster, if we went through fewer hits in the realtime

// cluster than the requested number of results.

return true;

}

// If we have enough hits, don't query the full archive cluster yet.

int numSuccessfulPartitions = realtimeResponse.getNumSuccessfulPartitions();

CollectorTerminationParams terminationParams =

request.getSearchQuery().getCollectorParams().getTerminationParams();

Preconditions.checkArgument(terminationParams.isSetMaxHitsToProcess());

int maxHits = terminationParams.getMaxHitsToProcess() \* numSuccessfulPartitions;

if (numHitsProcessed >= maxHits) {

return false;

}

// Check if there is a gap between the last result and the min status ID of current search.

// If the difference is larger than one day, then we can still get more tweets from the realtime

// cluster, so there's no need to query the full archive cluster just yet. If we don't check

// this, then we might end up with a big gap in the returned results.

int numReturnedResults = realtimeResponse.getSearchResults().getResultsSize();

if (numReturnedResults > 0) {

ThriftSearchResult lastResult =

realtimeResponse.getSearchResults().getResults().get(numReturnedResults - 1);

long lastResultTimeMillis = SnowflakeIdParser.getTimestampFromTweetId(lastResult.getId());

long minSearchedStatusID = realtimeResponse.getSearchResults().getMinSearchedStatusID();

long minSearchedStatusIDTimeMillis =

SnowflakeIdParser.getTimestampFromTweetId(minSearchedStatusID);

if (lastResultTimeMillis - minSearchedStatusIDTimeMillis > MILLIS\_IN\_ONE\_DAY) {

return false;

}

}

return true;

}

}