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

import java.util.Collection;

import java.util.List;

import java.util.concurrent.TimeUnit;

import com.google.common.collect.Collections2;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import com.twitter.search.common.util.earlybird.FacetsResultsUtils;

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

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

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

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

import com.twitter.util.Future;

/\*\*

\* Merger class to merge termstats EarlybirdResponse objects

\*/

public class TermStatisticsResponseMerger extends EarlybirdResponseMerger {

private static final Logger LOG = LoggerFactory.getLogger(TermStatisticsResponseMerger.class);

private static final SearchTimerStats TIMER =

SearchTimerStats.export("merge\_term\_stats", TimeUnit.NANOSECONDS, false, true);

private static final double SUCCESSFUL\_RESPONSE\_THRESHOLD = 0.9;

public TermStatisticsResponseMerger(EarlybirdRequestContext requestContext,

List<Future<EarlybirdResponse>> responses,

ResponseAccumulator mode) {

super(requestContext, responses, mode);

}

@Override

protected SearchTimerStats getMergedResponseTimer() {

return TIMER;

}

@Override

protected double getDefaultSuccessResponseThreshold() {

return SUCCESSFUL\_RESPONSE\_THRESHOLD;

}

@Override

protected EarlybirdResponse internalMerge(EarlybirdResponse termStatsResponse) {

ThriftTermStatisticsRequest termStatisticsRequest =

requestContext.getRequest().getTermStatisticsRequest();

Collection<EarlybirdResponse> termStatsResults =

Collections2.filter(accumulatedResponses.getSuccessResponses(),

earlybirdResponse -> earlybirdResponse.isSetTermStatisticsResults());

ThriftTermStatisticsResults results =

new ThriftTermResultsMerger(

termStatsResults,

termStatisticsRequest.getHistogramSettings())

.merge();

if (results.getTermResults().isEmpty()) {

final String line = "No results returned from any backend for term statistics request: {}";

// If the termstats request was not empty and we got empty results. log it as a warning

// otherwise log is as a debug.

if (termStatisticsRequest.getTermRequestsSize() > 0) {

LOG.warn(line, termStatisticsRequest);

} else {

LOG.debug(line, termStatisticsRequest);

}

}

termStatsResponse.setTermStatisticsResults(results);

termStatsResponse.setSearchResults(ThriftTermResultsMerger.mergeSearchStats(termStatsResults));

FacetsResultsUtils.fixNativePhotoUrl(results.getTermResults().values());

LOG.debug("TermStats call completed successfully: {}", termStatsResponse);

return termStatsResponse;

}

@Override

public boolean shouldEarlyTerminateTierMerge(int totalResultsFromSuccessfulShards,

boolean foundEarlyTermination) {

// To get accurate term stats, must never early terminate

return false;

}

}