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

import java.util.List;

import java.util.Map;

import com.google.common.base.Preconditions;

import com.google.common.collect.Maps;

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

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

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

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

/\*\*

\* Collection of EarlybirdResponses and associated stats to be merged.

\*/

public class AccumulatedResponses {

// The list of the successful responses from all earlybird futures. This does not include empty

// responses resulted from null requests.

private final List<EarlybirdResponse> successResponses;

// The list of the unsuccessful responses from all earlybird futures.

private final List<EarlybirdResponse> errorResponses;

// the list of max statusIds seen in each earlybird.

private final List<Long> maxIds;

// the list of min statusIds seen in each earlybird.

private final List<Long> minIds;

private final EarlyTerminationInfo mergedEarlyTerminationInfo;

private final boolean isMergingAcrossTiers;

private final PartitionCounts partitionCounts;

private final int numSearchedSegments;

public static final class PartitionCounts {

private final int numPartitions;

private final int numSuccessfulPartitions;

private final List<TierResponse> perTierResponse;

public PartitionCounts(int numPartitions, int numSuccessfulPartitions, List<TierResponse>

perTierResponse) {

this.numPartitions = numPartitions;

this.numSuccessfulPartitions = numSuccessfulPartitions;

this.perTierResponse = perTierResponse;

}

public int getNumPartitions() {

return numPartitions;

}

public int getNumSuccessfulPartitions() {

return numSuccessfulPartitions;

}

public List<TierResponse> getPerTierResponse() {

return perTierResponse;

}

}

/\*\*

\* Create AccumulatedResponses

\*/

public AccumulatedResponses(List<EarlybirdResponse> successResponses,

List<EarlybirdResponse> errorResponses,

List<Long> maxIds,

List<Long> minIds,

EarlyTerminationInfo mergedEarlyTerminationInfo,

boolean isMergingAcrossTiers,

PartitionCounts partitionCounts,

int numSearchedSegments) {

this.successResponses = successResponses;

this.errorResponses = errorResponses;

this.maxIds = maxIds;

this.minIds = minIds;

this.mergedEarlyTerminationInfo = mergedEarlyTerminationInfo;

this.isMergingAcrossTiers = isMergingAcrossTiers;

this.partitionCounts = partitionCounts;

this.numSearchedSegments = numSearchedSegments;

}

public List<EarlybirdResponse> getSuccessResponses() {

return successResponses;

}

public List<EarlybirdResponse> getErrorResponses() {

return errorResponses;

}

public List<Long> getMaxIds() {

return maxIds;

}

public List<Long> getMinIds() {

return minIds;

}

public EarlyTerminationInfo getMergedEarlyTerminationInfo() {

return mergedEarlyTerminationInfo;

}

public boolean foundError() {

return !errorResponses.isEmpty();

}

/\*\*

\* Tries to return a merged EarlybirdResponse that propagates as much information from the error

\* responses as possible.

\*

\* If all error responses have the same error response code, the merged response will have the

\* same error response code, and the debugString/debugInfo on the merged response will be set to

\* the debugString/debugInfo of one of the merged responses.

\*

\* If the error responses have at least 2 different response codes, TRANSIENT\_ERROR will be set

\* on the merged response. Also, we will look for the most common error response code, and will

\* propagate the debugString/debugInfo from an error response with that response code.

\*/

public EarlybirdResponse getMergedErrorResponse() {

Preconditions.checkState(!errorResponses.isEmpty());

// Find a response that has the most common error response code.

int maxCount = 0;

EarlybirdResponse errorResponseWithMostCommonErrorResponseCode = null;

Map<EarlybirdResponseCode, Integer> responseCodeCounts = Maps.newHashMap();

for (EarlybirdResponse errorResponse : errorResponses) {

EarlybirdResponseCode responseCode = errorResponse.getResponseCode();

Integer responseCodeCount = responseCodeCounts.get(responseCode);

if (responseCodeCount == null) {

responseCodeCount = 0;

}

++responseCodeCount;

responseCodeCounts.put(responseCode, responseCodeCount);

if (responseCodeCount > maxCount) {

errorResponseWithMostCommonErrorResponseCode = errorResponse;

}

}

// If all error responses have the same response code, set it on the merged response.

// Otherwise, set TRANSIENT\_ERROR on the merged response.

EarlybirdResponseCode mergedResponseCode = EarlybirdResponseCode.TRANSIENT\_ERROR;

if (responseCodeCounts.size() == 1) {

mergedResponseCode = responseCodeCounts.keySet().iterator().next();

}

EarlybirdResponse mergedResponse = new EarlybirdResponse()

.setResponseCode(mergedResponseCode);

// Propagate the debugString/debugInfo of the selected error response to the merged response.

Preconditions.checkNotNull(errorResponseWithMostCommonErrorResponseCode);

if (errorResponseWithMostCommonErrorResponseCode.isSetDebugString()) {

mergedResponse.setDebugString(errorResponseWithMostCommonErrorResponseCode.getDebugString());

}

if (errorResponseWithMostCommonErrorResponseCode.isSetDebugInfo()) {

mergedResponse.setDebugInfo(errorResponseWithMostCommonErrorResponseCode.getDebugInfo());

}

// Set the numPartitions and numPartitionsSucceeded on the mergedResponse

mergedResponse.setNumPartitions(partitionCounts.getNumPartitions());

mergedResponse.setNumSuccessfulPartitions(partitionCounts.getNumSuccessfulPartitions());

return mergedResponse;

}

public boolean isMergingAcrossTiers() {

return isMergingAcrossTiers;

}

public boolean isMergingPartitionsWithinATier() {

return !isMergingAcrossTiers;

}

public PartitionCounts getPartitionCounts() {

return partitionCounts;

}

public int getNumSearchedSegments() {

return numSearchedSegments;

}

}