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

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import com.google.common.base.Preconditions;

import com.google.common.collect.Lists;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

/\*\*

\* Generic MultiwayMergeCollector class for doing k-way merge of earlybird responses

\* that takes a comparator and returns a list of results sorted by the comparator.

\*/

public abstract class MultiwayMergeCollector<T> {

protected static final Logger LOG = LoggerFactory.getLogger(MultiwayMergeCollector.class);

private final Comparator<T> resultComparator;

private final int numResponsesToMerge;

private final List<T> results = Lists.newArrayList();

private int numResponsesAdded = 0;

/\*\*

\* Constructor that does multi way merge and takes in a custom predicate search result filter.

\*/

public MultiwayMergeCollector(int numResponses,

Comparator<T> comparator) {

Preconditions.checkNotNull(comparator);

this.resultComparator = comparator;

this.numResponsesToMerge = numResponses;

}

/\*\*

\* Add a single response from one partition, updates stats.

\*

\* @param response response from one partition

\*/

public final void addResponse(EarlybirdResponse response) {

// On prod, does it ever happen we receive more responses than numPartitions ?

Preconditions.checkArgument(numResponsesAdded++ < numResponsesToMerge,

String.format("Attempting to merge more than %d responses", numResponsesToMerge));

if (!isResponseValid(response)) {

return;

}

collectStats(response);

List<T> resultsFromResponse = collectResults(response);

if (resultsFromResponse != null && resultsFromResponse.size() > 0) {

results.addAll(resultsFromResponse);

}

}

/\*\*

\* Parse the EarlybirdResponse and retrieve list of results to be appended.

\*

\* @param response earlybird response from where results are extracted

\* @return resultsList to be appended

\*/

protected abstract List<T> collectResults(EarlybirdResponse response);

/\*\*

\* It is recommended that sub-class overrides this function to add custom logic to

\* collect more stat and call this base function.

\*/

protected void collectStats(EarlybirdResponse response) {

}

/\*\*

\* Get full list of results, after addResponse calls have been invoked.

\*

\* @return list of results extracted from all EarlybirdResponses that have been collected so far

\*/

protected final List<T> getResultsList() {

Collections.sort(results, resultComparator);

return results;

}

protected abstract boolean isResponseValid(EarlybirdResponse response);

}