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

import java.util.Collection;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import java.util.NavigableMap;

import javax.inject.Inject;

import javax.inject.Singleton;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.collect.ImmutableSortedMap;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

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

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

import com.twitter.search.earlybird.config.TierInfo;

import com.twitter.search.earlybird.config.TierInfoSource;

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

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

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

import com.twitter.snowflake.id.SnowflakeId;

import com.twitter.util.Future;

import com.twitter.util.FutureEventListener;

/\*\*

\* A filter to count the tier to which the oldest tweet in the results belong.

\*/

@Singleton

public class ResultTierCountFilter

extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

private static final String COUNTER\_PREFIX = "result\_tier\_count";

private final long firstTweetTimeSinceEpochSec;

private final NavigableMap<Long, SearchCounter> tierBuckets;

private final SearchCounter allCounter = SearchCounter.export(COUNTER\_PREFIX + "\_all");

private final SearchCounter noResultsCounter =

SearchCounter.export(COUNTER\_PREFIX + "\_no\_results");

@Inject

@SuppressWarnings("unused")

ResultTierCountFilter(TierInfoSource tierInfoSource) {

List<TierInfo> tierInfos = tierInfoSource.getTierInformation();

tierInfos.sort(Comparator.comparing(TierInfo::getDataStartDate));

firstTweetTimeSinceEpochSec = tierInfos.get(0).getServingRangeSinceTimeSecondsFromEpoch();

ImmutableSortedMap.Builder<Long, SearchCounter> builder = ImmutableSortedMap.naturalOrder();

Collections.reverse(tierInfos);

for (TierInfo tierInfo : tierInfos) {

SearchCounter searchCounter = SearchCounter.export(

String.format("%s\_%s", COUNTER\_PREFIX, tierInfo.getTierName()));

builder.put(tierInfo.getServingRangeSinceTimeSecondsFromEpoch(), searchCounter);

// export cumulative metrics to sum from the latest to a lower tier

Collection<SearchCounter> counters = builder.build().values();

SearchCustomGauge.export(

String.format("%s\_down\_to\_%s", COUNTER\_PREFIX, tierInfo.getTierName()),

() -> counters.stream()

.mapToLong(SearchCounter::get)

.sum());

}

tierBuckets = builder.build();

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequestContext context,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

return service.apply(context).addEventListener(

new FutureEventListener<EarlybirdResponse>() {

@Override

public void onFailure(Throwable cause) {

// do nothing

}

@Override

public void onSuccess(EarlybirdResponse response) {

record(response);

}

});

}

@VisibleForTesting

void record(EarlybirdResponse response) {

if (response.isSetSearchResults()) {

long minResultsStatusId = response.getSearchResults().getResults().stream()

.mapToLong(ThriftSearchResult::getId)

.min()

.orElse(-1);

getBucket(minResultsStatusId).increment();

}

allCounter.increment();

}

private SearchCounter getBucket(long statusId) {

if (statusId < 0) {

return noResultsCounter;

}

// If non-negative statusId is not a SnowflakeId, the tweet must have been created before

// Twepoch (2010-11-04T01:42:54Z) and thus belongs to full1.

long timeSinceEpochSec = firstTweetTimeSinceEpochSec;

if (SnowflakeId.isSnowflakeId(statusId)) {

timeSinceEpochSec = SnowflakeId.timeFromId(statusId).inSeconds();

}

return tierBuckets.floorEntry(timeSinceEpochSec).getValue();

}

}