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

import java.util.List;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

import java.util.concurrent.ConcurrentMap;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

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

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

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

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

import com.twitter.util.Future;

public class NamedMultiTermDisjunctionStatsFilter extends

SimpleFilter<EarlybirdRequest, EarlybirdResponse> {

private static final String STAT\_FORMAT = "named\_disjunction\_size\_client\_%s\_key\_%s";

// ClientID -> disjunction name -> operand count

private static final ConcurrentMap<String, ConcurrentMap<String, Percentile<Integer>>>

NAMED\_MULTI\_TERM\_DISJUNCTION\_IDS\_COUNT = new ConcurrentHashMap<>();

@Override

public Future<EarlybirdResponse> apply(EarlybirdRequest request,

Service<EarlybirdRequest, EarlybirdResponse> service) {

if (request.getSearchQuery().isSetNamedDisjunctionMap()) {

for (Map.Entry<String, List<Long>> entry

: request.getSearchQuery().getNamedDisjunctionMap().entrySet()) {

Map<String, Percentile<Integer>> statsForClient =

NAMED\_MULTI\_TERM\_DISJUNCTION\_IDS\_COUNT.computeIfAbsent(

request.getClientId(), clientId -> new ConcurrentHashMap<>());

Percentile<Integer> stats = statsForClient.computeIfAbsent(entry.getKey(),

keyName -> PercentileUtil.createPercentile(

String.format(STAT\_FORMAT, request.getClientId(), keyName)));

stats.record(entry.getValue().size());

}

}

return service.apply(request);

}

}