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

import java.util.concurrent.ConcurrentHashMap;

import java.util.concurrent.ConcurrentMap;

import javax.inject.Inject;

import com.google.common.annotations.VisibleForTesting;

import com.twitter.common.collections.Pair;

import com.twitter.common.util.Clock;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.clientstats.RequestCounters;

import com.twitter.search.common.clientstats.RequestCountersEventListener;

import com.twitter.search.common.decider.SearchDecider;

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

import com.twitter.search.common.util.FinagleUtil;

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

import com.twitter.search.earlybird.common.ClientIdUtil;

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

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

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

import com.twitter.util.Future;

/\*\* Tracks the number of queries we get from each client. \*/

public class ClientIdTrackingFilter extends SimpleFilter<EarlybirdRequest, EarlybirdResponse> {

// Be careful when changing the names of these stats or adding new ones: make sure that they have

// prefixes/suffixes that will allow us to group them in Viz, without pulling in other stats.

// For example, we'll probably have a Viz graph for client\_id\_tracker\_qps\_for\_client\_id\_\*\_all.

// So if you add a new stat named client\_id\_tracker\_qps\_for\_client\_id\_%s\_and\_new\_field\_%s\_all,

// then the graph will be grouping up the values from both stats, instead of grouping up the

// values only for client\_id\_tracker\_qps\_for\_client\_id\_%s\_all.

@VisibleForTesting

static final String QPS\_ALL\_STAT\_PATTERN = "client\_id\_tracker\_qps\_for\_%s\_all";

@VisibleForTesting

static final String QPS\_LOGGED\_IN\_STAT\_PATTERN = "client\_id\_tracker\_qps\_for\_%s\_logged\_in";

@VisibleForTesting

static final String QPS\_LOGGED\_OUT\_STAT\_PATTERN = "client\_id\_tracker\_qps\_for\_%s\_logged\_out";

static final String SUPERROOT\_REJECT\_REQUESTS\_WITH\_UNKNOWN\_FINAGLE\_ID =

"superroot\_reject\_requests\_with\_unknown\_finagle\_id";

static final String UNKNOWN\_FINAGLE\_ID\_DEBUG\_STRING = "Please specify a finagle client id.";

private final ConcurrentMap<String, RequestCounters> requestCountersByClientId =

new ConcurrentHashMap<>();

private final ConcurrentMap<Pair<String, String>, RequestCounters>

requestCountersByFinagleIdAndClientId = new ConcurrentHashMap<>();

private final Clock clock;

private final SearchDecider decider;

@Inject

public ClientIdTrackingFilter(SearchDecider decider) {

this(decider, Clock.SYSTEM\_CLOCK);

}

@VisibleForTesting

ClientIdTrackingFilter(SearchDecider decider, Clock clock) {

this.decider = decider;

this.clock = clock;

}

@Override

public Future<EarlybirdResponse> apply(EarlybirdRequest request,

Service<EarlybirdRequest, EarlybirdResponse> service) {

String clientId = ClientIdUtil.getClientIdFromRequest(request);

String finagleId = FinagleUtil.getFinagleClientName();

boolean isLoggedIn = ThriftSearchQueryUtil.requestInitiatedByLoggedInUser(request);

incrementCounters(clientId, finagleId, isLoggedIn);

if (decider.isAvailable(SUPERROOT\_REJECT\_REQUESTS\_WITH\_UNKNOWN\_FINAGLE\_ID)

&& finagleId.equals(FinagleUtil.UNKNOWN\_CLIENT\_NAME)) {

EarlybirdResponse response = new EarlybirdResponse(

EarlybirdResponseCode.QUOTA\_EXCEEDED\_ERROR, 0)

.setDebugString(UNKNOWN\_FINAGLE\_ID\_DEBUG\_STRING);

return Future.value(response);

}

RequestCounters clientCounters = getClientCounters(clientId);

RequestCountersEventListener<EarlybirdResponse> clientCountersEventListener =

new RequestCountersEventListener<>(

clientCounters, clock, EarlybirdSuccessfulResponseHandler.INSTANCE);

RequestCounters finagleIdAndClientCounters = getFinagleIdClientCounters(clientId, finagleId);

RequestCountersEventListener<EarlybirdResponse> finagleIdAndClientCountersEventListener =

new RequestCountersEventListener<>(

finagleIdAndClientCounters, clock, EarlybirdSuccessfulResponseHandler.INSTANCE);

return service.apply(request)

.addEventListener(clientCountersEventListener)

.addEventListener(finagleIdAndClientCountersEventListener);

}

// Returns the RequestCounters instance tracking the requests from the given client ID.

private RequestCounters getClientCounters(String clientId) {

RequestCounters clientCounters = requestCountersByClientId.get(clientId);

if (clientCounters == null) {

clientCounters = new RequestCounters(ClientIdUtil.formatClientId(clientId));

RequestCounters existingCounters =

requestCountersByClientId.putIfAbsent(clientId, clientCounters);

if (existingCounters != null) {

clientCounters = existingCounters;

}

}

return clientCounters;

}

// Returns the RequestCounters instance tracking the requests from the given client ID.

private RequestCounters getFinagleIdClientCounters(String clientId, String finagleId) {

Pair<String, String> clientKey = Pair.of(clientId, finagleId);

RequestCounters counters = requestCountersByFinagleIdAndClientId.get(clientKey);

if (counters == null) {

counters = new RequestCounters(ClientIdUtil.formatFinagleClientIdAndClientId(

finagleId, clientId));

RequestCounters existingCounters = requestCountersByFinagleIdAndClientId.putIfAbsent(

clientKey, counters);

if (existingCounters != null) {

counters = existingCounters;

}

}

return counters;

}

// Increments the correct counters, based on the given clientId, finagleId, and whether or not the

// request came from a logged in user.

private static void incrementCounters(String clientId, String finagleId, boolean isLoggedIn) {

String clientIdForStats = ClientIdUtil.formatClientId(clientId);

String finagleClientIdAndClientIdForStats =

ClientIdUtil.formatFinagleClientIdAndClientId(finagleId, clientId);

SearchCounter.export(String.format(QPS\_ALL\_STAT\_PATTERN, clientIdForStats)).increment();

SearchCounter.export(String.format(QPS\_ALL\_STAT\_PATTERN, finagleClientIdAndClientIdForStats))

.increment();

if (isLoggedIn) {

SearchCounter.export(String.format(QPS\_LOGGED\_IN\_STAT\_PATTERN, clientIdForStats)).increment();

SearchCounter.export(

String.format(QPS\_LOGGED\_IN\_STAT\_PATTERN, finagleClientIdAndClientIdForStats))

.increment();

} else {

SearchCounter.export(String.format(QPS\_LOGGED\_OUT\_STAT\_PATTERN, clientIdForStats))

.increment();

SearchCounter.export(

String.format(QPS\_LOGGED\_OUT\_STAT\_PATTERN, finagleClientIdAndClientIdForStats))

.increment();

}

}

}