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

import java.util.HashMap;

import java.util.Map;

import javax.annotation.Nullable;

import javax.inject.Inject;

import com.google.common.annotations.VisibleForTesting;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.constants.thriftjava.ThriftQuerySource;

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

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

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

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

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

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

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

import com.twitter.util.Future;

/\*\*

\* Rejects requests based on the query source of the request. Intended to be used at super-root

\* or archive-root. If used to reject client request at super-root, the client will get a response

\* with empty results and a REQUEST\_BLOCKED\_ERROR status code. If used at archive-root the client

\* will get a response which might contain some results from realtime and protected and the status

\* code of the response will depend on how super-root combines responses from the three downstream

\* roots.

\*/

public class RejectRequestsByQuerySourceFilter extends

SimpleFilter<EarlybirdRequest, EarlybirdResponse> {

@VisibleForTesting

protected static final String NUM\_REJECTED\_REQUESTS\_STAT\_NAME\_PATTERN =

"num\_root\_%s\_rejected\_requests\_with\_query\_source\_%s";

@VisibleForTesting

protected static final String REJECT\_REQUESTS\_DECIDER\_KEY\_PATTERN =

"root\_%s\_reject\_requests\_with\_query\_source\_%s";

private final Map<ThriftQuerySource, SearchRateCounter> rejectedRequestsCounterPerQuerySource =

new HashMap<>();

private final Map<ThriftQuerySource, String> rejectRequestsDeciderKeyPerQuerySource =

new HashMap<>();

private final SearchDecider searchDecider;

@Inject

public RejectRequestsByQuerySourceFilter(

@Nullable EarlybirdCluster cluster,

SearchDecider searchDecider) {

this.searchDecider = searchDecider;

String clusterName = cluster != null

? cluster.getNameForStats()

: EarlybirdCluster.SUPERROOT.getNameForStats();

for (ThriftQuerySource querySource : ThriftQuerySource.values()) {

String querySourceName = querySource.name().toLowerCase();

rejectedRequestsCounterPerQuerySource.put(querySource,

SearchRateCounter.export(

String.format(

NUM\_REJECTED\_REQUESTS\_STAT\_NAME\_PATTERN, clusterName, querySourceName)));

rejectRequestsDeciderKeyPerQuerySource.put(querySource,

String.format(

REJECT\_REQUESTS\_DECIDER\_KEY\_PATTERN, clusterName, querySourceName));

}

}

@Override

public Future<EarlybirdResponse> apply(EarlybirdRequest request,

Service<EarlybirdRequest, EarlybirdResponse> service) {

ThriftQuerySource querySource = request.isSetQuerySource()

? request.getQuerySource()

: ThriftQuerySource.UNKNOWN;

String deciderKey = rejectRequestsDeciderKeyPerQuerySource.get(querySource);

if (searchDecider.isAvailable(deciderKey)) {

rejectedRequestsCounterPerQuerySource.get(querySource).increment();

return Future.value(getRejectedRequestResponse(querySource, deciderKey));

}

return service.apply(request);

}

private static EarlybirdResponse getRejectedRequestResponse(

ThriftQuerySource querySource, String deciderKey) {

return new EarlybirdResponse(EarlybirdResponseCode.REQUEST\_BLOCKED\_ERROR, 0)

.setSearchResults(new ThriftSearchResults())

.setDebugString(String.format(

"Request with query source %s is blocked by decider %s", querySource, deciderKey));

}

}