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

import java.util.concurrent.TimeUnit;

import javax.inject.Inject;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.root.RequestSuccessStats;

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

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;

import com.twitter.util.FutureEventListener;

import static com.twitter.search.common.util.earlybird.EarlybirdResponseUtil.responseConsideredFailed;

/\*\*

\* Records cancellations, timeouts, and failures for requests that do not go through

\* ScatterGatherService (which also updates these stats, but for different requests).

\*/

public class RequestSuccessStatsFilter

extends SimpleFilter<EarlybirdRequest, EarlybirdResponse> {

private final RequestSuccessStats stats;

@Inject

RequestSuccessStatsFilter(RequestSuccessStats stats) {

this.stats = stats;

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequest request,

Service<EarlybirdRequest, EarlybirdResponse> service) {

final long startTime = System.nanoTime();

return service.apply(request).addEventListener(

new FutureEventListener<EarlybirdResponse>() {

@Override

public void onSuccess(EarlybirdResponse response) {

boolean success = true;

if (response.getResponseCode() == EarlybirdResponseCode.CLIENT\_CANCEL\_ERROR) {

success = false;

stats.getCancelledRequestCount().increment();

} else if (response.getResponseCode() == EarlybirdResponseCode.SERVER\_TIMEOUT\_ERROR) {

success = false;

stats.getTimedoutRequestCount().increment();

} else if (responseConsideredFailed(response.getResponseCode())) {

success = false;

stats.getErroredRequestCount().increment();

}

long latencyNanos = System.nanoTime() - startTime;

stats.getRequestLatencyStats().requestComplete(

TimeUnit.NANOSECONDS.toMillis(latencyNanos), 0, success);

}

@Override

public void onFailure(Throwable cause) {

long latencyNanos = System.nanoTime() - startTime;

stats.getRequestLatencyStats().requestComplete(

TimeUnit.NANOSECONDS.toMillis(latencyNanos), 0, false);

if (FinagleUtil.isCancelException(cause)) {

stats.getCancelledRequestCount().increment();

} else if (FinagleUtil.isTimeoutException(cause)) {

stats.getTimedoutRequestCount().increment();

} else {

stats.getErroredRequestCount().increment();

}

}

});

}

}