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

import java.util.concurrent.TimeUnit;

import javax.inject.Inject;

import scala.Option;

import com.google.common.base.Preconditions;

import com.google.common.cache.CacheBuilder;

import com.google.common.cache.CacheLoader;

import com.google.common.cache.LoadingCache;

import com.twitter.common.util.Clock;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.finagle.context.Contexts$;

import com.twitter.finagle.context.Deadline;

import com.twitter.finagle.context.Deadline$;

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

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

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\_root.common.EarlybirdRequestContext;

import com.twitter.util.Future;

/\*\*

\* A filter for comparing the request deadline (set in the finagle request context) with the request

\* timeout, as set in the EarlybirdRequest.

\*

\* Tracks stats per client, for (1) requests where the request deadline is set to expire before the

\* EarlybirdRequest timeout, and also (2) requests where the deadline allows enough time for the

\* EarlybirdRequest timeout to kick in.

\*/

public class DeadlineTimeoutStatsFilter

extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

// All stats maps below are per client id, keyed by the client id.

private final LoadingCache<String, SearchCounter> requestTimeoutNotSetStats;

private final LoadingCache<String, SearchCounter> finagleDeadlineNotSetStats;

private final LoadingCache<String, SearchCounter> finagleDeadlineAndRequestTimeoutNotSetStats;

private final LoadingCache<String, SearchTimerStats> requestTimeoutStats;

private final LoadingCache<String, SearchTimerStats> finagleDeadlineStats;

private final LoadingCache<String, SearchTimerStats> deadlineLargerStats;

private final LoadingCache<String, SearchTimerStats> deadlineSmallerStats;

@Inject

public DeadlineTimeoutStatsFilter(Clock clock) {

this.requestTimeoutNotSetStats = CacheBuilder.newBuilder().build(

new CacheLoader<String, SearchCounter>() {

public SearchCounter load(String clientId) {

return SearchCounter.export(

"deadline\_for\_client\_id\_" + clientId + "\_request\_timeout\_not\_set");

}

});

this.finagleDeadlineNotSetStats = CacheBuilder.newBuilder().build(

new CacheLoader<String, SearchCounter>() {

public SearchCounter load(String clientId) {

return SearchCounter.export(

"deadline\_for\_client\_id\_" + clientId + "\_finagle\_deadline\_not\_set");

}

});

this.finagleDeadlineAndRequestTimeoutNotSetStats = CacheBuilder.newBuilder().build(

new CacheLoader<String, SearchCounter>() {

public SearchCounter load(String clientId) {

return SearchCounter.export(

"deadline\_for\_client\_id\_" + clientId

+ "\_finagle\_deadline\_and\_request\_timeout\_not\_set");

}

});

this.requestTimeoutStats = CacheBuilder.newBuilder().build(

new CacheLoader<String, SearchTimerStats>() {

public SearchTimerStats load(String clientId) {

return SearchTimerStats.export(

"deadline\_for\_client\_id\_" + clientId + "\_request\_timeout",

TimeUnit.MILLISECONDS,

false,

true,

clock);

}

});

this.finagleDeadlineStats = CacheBuilder.newBuilder().build(

new CacheLoader<String, SearchTimerStats>() {

public SearchTimerStats load(String clientId) {

return SearchTimerStats.export(

"deadline\_for\_client\_id\_" + clientId + "\_finagle\_deadline",

TimeUnit.MILLISECONDS,

false,

true,

clock);

}

});

this.deadlineLargerStats = CacheBuilder.newBuilder().build(

new CacheLoader<String, SearchTimerStats>() {

public SearchTimerStats load(String clientId) {

return SearchTimerStats.export(

"deadline\_for\_client\_id\_" + clientId

+ "\_finagle\_deadline\_larger\_than\_request\_timeout",

TimeUnit.MILLISECONDS,

false,

true,

clock

);

}

});

this.deadlineSmallerStats = CacheBuilder.newBuilder().build(

new CacheLoader<String, SearchTimerStats>() {

public SearchTimerStats load(String clientId) {

return SearchTimerStats.export(

"deadline\_for\_client\_id\_" + clientId

+ "\_finagle\_deadline\_smaller\_than\_request\_timeout",

TimeUnit.MILLISECONDS,

false,

true,

clock

);

}

});

}

@Override

public Future<EarlybirdResponse> apply(

EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

EarlybirdRequest request = requestContext.getRequest();

String clientId = ClientIdUtil.getClientIdFromRequest(request);

long requestTimeoutMillis = getRequestTimeout(request);

Option<Deadline> deadline = Contexts$.MODULE$.broadcast().get(Deadline$.MODULE$);

// Tracking per-client timeouts specified in the EarlybirdRequest.

if (requestTimeoutMillis > 0) {

requestTimeoutStats.getUnchecked(clientId).timerIncrement(requestTimeoutMillis);

} else {

requestTimeoutNotSetStats.getUnchecked(clientId).increment();

}

// How much time does this request have, from its deadline start, to the effective deadline.

if (deadline.isDefined()) {

long deadlineEndTimeMillis = deadline.get().deadline().inMillis();

long deadlineStartTimeMillis = deadline.get().timestamp().inMillis();

long finagleDeadlineTimeMillis = deadlineEndTimeMillis - deadlineStartTimeMillis;

finagleDeadlineStats.getUnchecked(clientId).timerIncrement(finagleDeadlineTimeMillis);

} else {

finagleDeadlineNotSetStats.getUnchecked(clientId).increment();

}

// Explicitly track when both are not set.

if (requestTimeoutMillis <= 0 && deadline.isEmpty()) {

finagleDeadlineAndRequestTimeoutNotSetStats.getUnchecked(clientId).increment();

}

// If both timeout and the deadline are set, track how much over / under we are, when

// comparing the deadline, and the EarlybirdRequest timeout.

if (requestTimeoutMillis > 0 && deadline.isDefined()) {

long deadlineEndTimeMillis = deadline.get().deadline().inMillis();

Preconditions.checkState(request.isSetClientRequestTimeMs(),

"Expect ClientRequestTimeFilter to always set the clientRequestTimeMs field. Request: %s",

request);

long requestStartTimeMillis = request.getClientRequestTimeMs();

long requestEndTimeMillis = requestStartTimeMillis + requestTimeoutMillis;

long deadlineDiffMillis = deadlineEndTimeMillis - requestEndTimeMillis;

if (deadlineDiffMillis >= 0) {

deadlineLargerStats.getUnchecked(clientId).timerIncrement(deadlineDiffMillis);

} else {

// Track "deadline is smaller" as positive values.

deadlineSmallerStats.getUnchecked(clientId).timerIncrement(-deadlineDiffMillis);

}

}

return service.apply(requestContext);

}

private long getRequestTimeout(EarlybirdRequest request) {

if (request.isSetSearchQuery()

&& request.getSearchQuery().isSetCollectorParams()

&& request.getSearchQuery().getCollectorParams().isSetTerminationParams()

&& request.getSearchQuery().getCollectorParams().getTerminationParams().isSetTimeoutMs()) {

return request.getSearchQuery().getCollectorParams().getTerminationParams().getTimeoutMs();

} else if (request.isSetTimeoutMs()) {

return request.getTimeoutMs();

} else {

return -1;

}

}

}