package com.twitter.tweetypie

package service

package observer

import com.twitter.servo.exception.thriftscala.ClientError

import com.twitter.tweetypie.thriftscala.GetTweetOptions

import com.twitter.tweetypie.thriftscala.GetTweetResult

import com.twitter.tweetypie.thriftscala.GetTweetsRequest

private[service] object GetTweetsObserver {

type Type = ObserveExchange[GetTweetsRequest, Seq[GetTweetResult]]

def observeExchange(stats: StatsReceiver): Effect[Type] = {

val resultStateStats = ResultStateStats(stats)

Effect {

case (request, response) =>

response match {

case Return(xs) =>

xs.foreach {

case result if Observer.successStatusStates(result.tweetState) =>

resultStateStats.success()

case \_ =>

resultStateStats.failed()

}

case Throw(ClientError(\_)) =>

resultStateStats.success(request.tweetIds.size)

case Throw(\_) =>

resultStateStats.failed(request.tweetIds.size)

}

}

}

def observeResults(stats: StatsReceiver, byClient: Boolean): Effect[Seq[GetTweetResult]] =

countStates(stats).also(countTweetReadAttributes(stats, byClient))

def observeRequest(stats: StatsReceiver, byClient: Boolean): Effect[GetTweetsRequest] = {

val requestSizeStat = stats.stat("request\_size")

val optionsScope = stats.scope("options")

val languageScope = optionsScope.scope("language")

val includeSourceTweetCounter = optionsScope.counter("source\_tweet")

val includeQuotedTweetCounter = optionsScope.counter("quoted\_tweet")

val includePerspectiveCounter = optionsScope.counter("perspective")

val includeConversationMutedCounter = optionsScope.counter("conversation\_muted")

val includePlacesCounter = optionsScope.counter("places")

val includeCardsCounter = optionsScope.counter("cards")

val includeRetweetCountsCounter = optionsScope.counter("retweet\_counts")

val includeReplyCountsCounter = optionsScope.counter("reply\_counts")

val includeFavoriteCountsCounter = optionsScope.counter("favorite\_counts")

val includeQuoteCountsCounter = optionsScope.counter("quote\_counts")

val bypassVisibilityFilteringCounter = optionsScope.counter("bypass\_visibility\_filtering")

val excludeReportedCounter = optionsScope.counter("exclude\_reported")

val cardsPlatformKeyScope = optionsScope.scope("cards\_platform\_key")

val extensionsArgsCounter = optionsScope.counter("extensions\_args")

val doNotCacheCounter = optionsScope.counter("do\_not\_cache")

val additionalFieldsScope = optionsScope.scope("additional\_fields")

val safetyLevelScope = optionsScope.scope("safety\_level")

val includeProfileGeoEnrichment = optionsScope.counter("profile\_geo\_enrichment")

val includeMediaAdditionalMetadata = optionsScope.counter("media\_additional\_metadata")

val simpleQuotedTweet = optionsScope.counter("simple\_quoted\_tweet")

val forUserIdCounter = optionsScope.counter("for\_user\_id")

def includesPerspectivals(options: GetTweetOptions) =

options.includePerspectivals && options.forUserId.nonEmpty

Effect {

case GetTweetsRequest(tweetIds, \_, Some(options), \_) =>

requestSizeStat.add(tweetIds.size)

if (!byClient) languageScope.counter(options.languageTag).incr()

if (options.includeSourceTweet) includeSourceTweetCounter.incr()

if (options.includeQuotedTweet) includeQuotedTweetCounter.incr()

if (includesPerspectivals(options)) includePerspectiveCounter.incr()

if (options.includeConversationMuted) includeConversationMutedCounter.incr()

if (options.includePlaces) includePlacesCounter.incr()

if (options.includeCards) includeCardsCounter.incr()

if (options.includeRetweetCount) includeRetweetCountsCounter.incr()

if (options.includeReplyCount) includeReplyCountsCounter.incr()

if (options.includeFavoriteCount) includeFavoriteCountsCounter.incr()

if (options.includeQuoteCount) includeQuoteCountsCounter.incr()

if (options.bypassVisibilityFiltering) bypassVisibilityFilteringCounter.incr()

if (options.excludeReported) excludeReportedCounter.incr()

if (options.extensionsArgs.nonEmpty) extensionsArgsCounter.incr()

if (options.doNotCache) doNotCacheCounter.incr()

if (options.includeProfileGeoEnrichment) includeProfileGeoEnrichment.incr()

if (options.includeMediaAdditionalMetadata) includeMediaAdditionalMetadata.incr()

if (options.simpleQuotedTweet) simpleQuotedTweet.incr()

if (options.forUserId.nonEmpty) forUserIdCounter.incr()

if (!byClient) {

options.cardsPlatformKey.foreach { cardsPlatformKey =>

cardsPlatformKeyScope.counter(cardsPlatformKey).incr()

}

}

options.additionalFieldIds.foreach { id =>

additionalFieldsScope.counter(id.toString).incr()

}

options.safetyLevel.foreach { level => safetyLevelScope.counter(level.toString).incr() }

}

}

/\*\*

\* We count the number of times each tweet state is returned as a

\* general measure of the health of TweetyPie. partial and not\_found

\* tweet states should be close to zero.

\*/

private def countStates(stats: StatsReceiver): Effect[Seq[GetTweetResult]] = {

val state = Observer.observeStatusStates(stats)

Effect { results => results.foreach { tweetResult => state(tweetResult.tweetState) } }

}

private def countTweetReadAttributes(

stats: StatsReceiver,

byClient: Boolean

): Effect[Seq[GetTweetResult]] = {

val tweetObserver = Observer.countTweetAttributes(stats, byClient)

Effect { results =>

results.foreach { tweetResult => tweetResult.tweet.foreach(tweetObserver) }

}

}

}