package com.twitter.timelineranker.server

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.thrift.ClientId

import com.twitter.logging.Logger

import com.twitter.timelineranker.model.\_

import com.twitter.timelines.warmup.TwitterServerWarmup

import com.twitter.timelineservice.model.TimelineId

import com.twitter.timelineservice.model.core.TimelineKind

import com.twitter.timelineranker.config.TimelineRankerConstants

import com.twitter.timelineranker.thriftscala.{TimelineRanker => ThriftTimelineRanker}

import com.twitter.util.Future

import com.twitter.util.Duration

object Warmup {

val WarmupForwardingTime: Duration = 25.seconds

}

class Warmup(

localInstance: TimelineRanker,

forwardingClient: ThriftTimelineRanker.MethodPerEndpoint,

override val logger: Logger)

extends TwitterServerWarmup {

override val WarmupClientId: ClientId = ClientId(TimelineRankerConstants.WarmupClientName)

override val NumWarmupRequests = 20

override val MinSuccessfulRequests = 10

private[this] val warmupUserId = Math.abs(scala.util.Random.nextLong())

private[server] val reverseChronQuery = ReverseChronTimelineQuery(

id = new TimelineId(warmupUserId, TimelineKind.home),

maxCount = Some(20),

range = Some(TweetIdRange.default)

).toThrift

private[server] val recapQuery = RecapQuery(

userId = warmupUserId,

maxCount = Some(20),

range = Some(TweetIdRange.default)

).toThriftRecapQuery

override def sendSingleWarmupRequest(): Future[Unit] = {

val localWarmups = Seq(

localInstance.getTimelines(Seq(reverseChronQuery)),

localInstance.getRecycledTweetCandidates(Seq(recapQuery))

)

// send forwarding requests but ignore failures

forwardingClient.getTimelines(Seq(reverseChronQuery)).unit.handle {

case e => logger.warning(e, "fowarding request failed")

}

Future.join(localWarmups).unit

}

}