package com.twitter.simclustersann

import com.twitter.inject.Logging

import com.twitter.inject.utils.Handler

import javax.inject.Inject

import scala.util.control.NonFatal

import com.google.common.util.concurrent.RateLimiter

import com.twitter.conversions.DurationOps.richDurationFromInt

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.simclusters\_v2.common.ClusterId

import com.twitter.simclusters\_v2.common.TweetId

import com.twitter.storehaus.ReadableStore

import com.twitter.util.Await

import com.twitter.util.ExecutorServiceFuturePool

import com.twitter.util.Future

class SimclustersAnnWarmupHandler @Inject() (

clusterTweetCandidatesStore: ReadableStore[ClusterId, Seq[(TweetId, Double)]],

futurePool: ExecutorServiceFuturePool,

rateLimiter: RateLimiter,

statsReceiver: StatsReceiver)

extends Handler

with Logging {

private val stats = statsReceiver.scope(this.getClass.getName)

private val scopedStats = stats.scope("fetchFromCache")

private val clusters = scopedStats.counter("clusters")

private val fetchedKeys = scopedStats.counter("keys")

private val failures = scopedStats.counter("failures")

private val success = scopedStats.counter("success")

private val SimclustersNumber = 144428

override def handle(): Unit = {

try {

val clusterIds = List.range(1, SimclustersNumber)

val futures: Seq[Future[Unit]] = clusterIds

.map { clusterId =>

clusters.incr()

futurePool {

rateLimiter.acquire()

Await.result(

clusterTweetCandidatesStore

.get(clusterId)

.onSuccess { \_ =>

success.incr()

}

.handle {

case NonFatal(e) =>

failures.incr()

},

timeout = 10.seconds

)

fetchedKeys.incr()

}

}

Await.result(Future.collect(futures), timeout = 10.minutes)

} catch {

case NonFatal(e) => error(e.getMessage, e)

} finally {

try {

futurePool.executor.shutdown()

} catch {

case NonFatal(\_) =>

}

info("Warmup done.")

}

}

}