package com.twitter.frigate.pushservice.store

import com.twitter.cr\_mixer.thriftscala.CrMixer

import com.twitter.cr\_mixer.thriftscala.CrMixerTweetRequest

import com.twitter.cr\_mixer.thriftscala.CrMixerTweetResponse

import com.twitter.cr\_mixer.thriftscala.FrsTweetRequest

import com.twitter.cr\_mixer.thriftscala.FrsTweetResponse

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.Stat

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.util.Future

/\*\*

\* Store to get content recs from content recommender.

\*/

case class CrMixerTweetStore(

crMixer: CrMixer.MethodPerEndpoint

)(

implicit statsReceiver: StatsReceiver = NullStatsReceiver) {

private val requestsCounter = statsReceiver.counter("requests")

private val successCounter = statsReceiver.counter("success")

private val failuresCounter = statsReceiver.counter("failures")

private val nonEmptyCounter = statsReceiver.counter("non\_empty")

private val emptyCounter = statsReceiver.counter("empty")

private val failuresScope = statsReceiver.scope("failures")

private val latencyStat = statsReceiver.stat("latency")

private def updateStats[T](f: => Future[Option[T]]): Future[Option[T]] = {

requestsCounter.incr()

Stat

.timeFuture(latencyStat)(f)

.onSuccess { r =>

if (r.isDefined) nonEmptyCounter.incr() else emptyCounter.incr()

successCounter.incr()

}

.onFailure { e =>

{

failuresCounter.incr()

failuresScope.counter(e.getClass.getName).incr()

}

}

}

def getTweetRecommendations(

request: CrMixerTweetRequest

): Future[Option[CrMixerTweetResponse]] = {

updateStats(crMixer.getTweetRecommendations(request).map { response =>

Some(response)

})

}

def getFRSTweetCandidates(request: FrsTweetRequest): Future[Option[FrsTweetResponse]] = {

updateStats(crMixer.getFrsBasedTweetRecommendations(request).map { response =>

Some(response)

})

}

}