package com.twitter.timelineranker.recap\_author

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.service.RetryPolicy

import com.twitter.timelineranker.config.RequestScopes

import com.twitter.timelineranker.config.RuntimeConfiguration

import com.twitter.timelineranker.repository.CandidatesRepositoryBuilder

import com.twitter.timelineranker.visibility.SgsFollowGraphDataFields

import com.twitter.search.earlybird.thriftscala.EarlybirdService

import com.twitter.timelines.util.stats.RequestScope

import com.twitter.util.Duration

class RecapAuthorRepositoryBuilder(config: RuntimeConfiguration)

extends CandidatesRepositoryBuilder(config) {

override val clientSubId = "recap\_by\_author"

override val requestScope: RequestScope = RequestScopes.RecapAuthorSource

override val followGraphDataFieldsToFetch: SgsFollowGraphDataFields.ValueSet =

SgsFollowGraphDataFields.ValueSet(

SgsFollowGraphDataFields.FollowedUserIds,

SgsFollowGraphDataFields.MutuallyFollowingUserIds,

SgsFollowGraphDataFields.MutedUserIds

)

/\*\*

\* Budget for processing within the search root cluster for the recap\_by\_author query.

\*/

override val searchProcessingTimeout: Duration = 250.milliseconds

private val EarlybirdTimeout = 650.milliseconds

private val EarlybirdRequestTimeout = 600.milliseconds

private val EarlybirdRealtimeCGTimeout = 650.milliseconds

private val EarlybirdRealtimeCGRequestTimeout = 600.milliseconds

/\*\*

\* TLM -> TLR timeout is 1s for candidate retrieval, so make the finagle TLR -> EB timeout

\* a bit shorter than 1s.

\*/

override def earlybirdClient(scope: String): EarlybirdService.MethodPerEndpoint =

config.underlyingClients.createEarlybirdClient(

scope = scope,

requestTimeout = EarlybirdRequestTimeout,

// timeout is slight less than timelineranker client timeout in timelinemixer

timeout = EarlybirdTimeout,

retryPolicy = RetryPolicy.Never

)

/\*\* The RealtimeCG clients below are only used for the Earlybird Cluster Migration \*/

private def earlybirdRealtimeCGClient(scope: String): EarlybirdService.MethodPerEndpoint =

config.underlyingClients.createEarlybirdRealtimeCgClient(

scope = scope,

requestTimeout = EarlybirdRealtimeCGRequestTimeout,

timeout = EarlybirdRealtimeCGTimeout,

retryPolicy = RetryPolicy.Never

)

private val realtimeCGClientSubId = "realtime\_cg\_recap\_by\_author"

private lazy val searchRealtimeCGClient =

newSearchClient(earlybirdRealtimeCGClient, clientId = realtimeCGClientSubId)

def apply(): RecapAuthorRepository = {

val recapAuthorSource = new RecapAuthorSource(

gizmoduckClient,

searchClient,

tweetyPieLowQoSClient,

userMetadataClient,

followGraphDataProvider, // Used to early-enforce visibility filtering, even though authorIds is part of query

config.underlyingClients.contentFeaturesCache,

clientFactories.visibilityEnforcerFactory.apply(

VisibilityRules,

RequestScopes.RecapAuthorSource

),

config.statsReceiver

)

val recapAuthorRealtimeCGSource = new RecapAuthorSource(

gizmoduckClient,

searchRealtimeCGClient,

tweetyPieLowQoSClient,

userMetadataClient,

followGraphDataProvider, // Used to early-enforce visibility filtering, even though authorIds is part of query

config.underlyingClients.contentFeaturesCache,

clientFactories.visibilityEnforcerFactory.apply(

VisibilityRules,

RequestScopes.RecapAuthorSource

),

config.statsReceiver.scope("replacementRealtimeCG")

)

new RecapAuthorRepository(recapAuthorSource, recapAuthorRealtimeCGSource)

}

}