package com.twitter.cr\_mixer.module

import com.google.inject.Provides

import com.google.inject.Singleton

import com.google.inject.name.Named

import com.twitter.conversions.DurationOps.\_

import com.twitter.cr\_mixer.model.ModuleNames

import com.twitter.cr\_mixer.similarity\_engine.SimilarityEngine.keyHasher

import com.twitter.finagle.memcached.{Client => MemcachedClient}

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.store.strato.StratoFetchableStore

import com.twitter.hermit.store.common.ObservedCachedReadableStore

import com.twitter.hermit.store.common.ObservedMemcachedReadableStore

import com.twitter.hermit.store.common.ObservedReadableStore

import com.twitter.inject.TwitterModule

import com.twitter.relevance\_platform.common.injection.LZ4Injection

import com.twitter.relevance\_platform.common.injection.SeqObjectInjection

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.client.Client

import com.twitter.topic\_recos.thriftscala.TopicTopTweets

import com.twitter.topic\_recos.thriftscala.TopicTweet

import com.twitter.topic\_recos.thriftscala.TopicTweetPartitionFlatKey

/\*\*

\* Strato store that wraps the topic top tweets pipeline indexed from a Summingbird job

\*/

object SkitStratoStoreModule extends TwitterModule {

val column = "recommendations/topic\_recos/topicTopTweets"

@Provides

@Singleton

@Named(ModuleNames.SkitStratoStoreName)

def providesSkitStratoStore(

@Named(ModuleNames.UnifiedCache) crMixerUnifiedCacheClient: MemcachedClient,

stratoClient: Client,

statsReceiver: StatsReceiver

): ReadableStore[TopicTweetPartitionFlatKey, Seq[TopicTweet]] = {

val skitStore = ObservedReadableStore(

StratoFetchableStore

.withUnitView[TopicTweetPartitionFlatKey, TopicTopTweets](stratoClient, column))(

statsReceiver.scope(ModuleNames.SkitStratoStoreName)).mapValues { topicTopTweets =>

topicTopTweets.topTweets

}

val memCachedStore = ObservedMemcachedReadableStore

.fromCacheClient(

backingStore = skitStore,

cacheClient = crMixerUnifiedCacheClient,

ttl = 10.minutes

)(

valueInjection = LZ4Injection.compose(SeqObjectInjection[TopicTweet]()),

statsReceiver = statsReceiver.scope("memcached\_skit\_store"),

keyToString = { k => s"skit:${keyHasher.hashKey(k.toString.getBytes)}" }

)

ObservedCachedReadableStore.from[TopicTweetPartitionFlatKey, Seq[TopicTweet]](

memCachedStore,

ttl = 5.minutes,

maxKeys = 100000, // ~150MB max

cacheName = "skit\_in\_memory\_cache",

windowSize = 10000L

)(statsReceiver.scope("skit\_in\_memory\_cache"))

}

}