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.inject.TwitterModule

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

import com.twitter.conversions.DurationOps.\_

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

import com.twitter.cr\_mixer.param.decider.CrMixerDecider

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.storage.client.manhattan.kv.ManhattanKVClientMtlsParams

import com.twitter.storehaus.ReadableStore

import com.twitter.storehaus\_internal.manhattan.Apollo

import com.twitter.storehaus\_internal.manhattan.ManhattanRO

import com.twitter.storehaus\_internal.manhattan.ManhattanROConfig

import com.twitter.storehaus\_internal.util.ApplicationID

import com.twitter.storehaus\_internal.util.DatasetName

import com.twitter.storehaus\_internal.util.HDFSPath

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.cr\_mixer.param.decider.DeciderKey

import com.twitter.hermit.store.common.DeciderableReadableStore

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

import com.twitter.wtf.candidate.thriftscala.CandidateSeq

object RealGraphStoreMhModule extends TwitterModule {

@Provides

@Singleton

@Named(ModuleNames.RealGraphInStore)

def providesRealGraphStoreMh(

decider: CrMixerDecider,

statsReceiver: StatsReceiver,

manhattanKVClientMtlsParams: ManhattanKVClientMtlsParams,

@Named(ModuleNames.UnifiedCache) crMixerUnifiedCacheClient: MemcachedClient,

): ReadableStore[UserId, CandidateSeq] = {

implicit val valueCodec = new BinaryScalaCodec(CandidateSeq)

val underlyingStore = ManhattanRO

.getReadableStoreWithMtls[UserId, CandidateSeq](

ManhattanROConfig(

HDFSPath(""),

ApplicationID("cr\_mixer\_apollo"),

DatasetName("real\_graph\_scores\_apollo"),

Apollo),

manhattanKVClientMtlsParams

)

val memCachedStore = ObservedMemcachedReadableStore

.fromCacheClient(

backingStore = underlyingStore,

cacheClient = crMixerUnifiedCacheClient,

ttl = 24.hours,

)(

valueInjection = valueCodec,

statsReceiver = statsReceiver.scope("memCachedUserRealGraphMh"),

keyToString = { k: UserId => s"uRGraph/$k" }

)

DeciderableReadableStore(

memCachedStore,

decider.deciderGateBuilder.idGate(DeciderKey.enableRealGraphMhStoreDeciderKey),

statsReceiver.scope("RealGraphMh")

)

}

}