package com.twitter.tsp.modules

import com.google.inject.Module

import com.google.inject.Provides

import com.google.inject.Singleton

import com.twitter.conversions.DurationOps.\_

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

import com.twitter.finagle.stats.StatsReceiver

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.simclusters\_v2.common.TweetId

import com.twitter.simclusters\_v2.thriftscala.Score

import com.twitter.simclusters\_v2.thriftscala.ScoreId

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.client.{Client => StratoClient}

import com.twitter.tsp.stores.SemanticCoreAnnotationStore

import com.twitter.tsp.stores.TopicSocialProofStore

import com.twitter.tsp.stores.TopicSocialProofStore.TopicSocialProof

import com.twitter.tsp.utils.LZ4Injection

import com.twitter.tsp.utils.SeqObjectInjection

object TopicSocialProofStoreModule extends TwitterModule {

override def modules: Seq[Module] = Seq(UnifiedCacheClient)

@Provides

@Singleton

def providesTopicSocialProofStore(

representationScorerStore: ReadableStore[ScoreId, Score],

statsReceiver: StatsReceiver,

stratoClient: StratoClient,

tspUnifiedCacheClient: MemClient,

): ReadableStore[TopicSocialProofStore.Query, Seq[TopicSocialProof]] = {

val semanticCoreAnnotationStore: ReadableStore[TweetId, Seq[

SemanticCoreAnnotationStore.TopicAnnotation

]] = ObservedReadableStore(

SemanticCoreAnnotationStore(SemanticCoreAnnotationStore.getStratoStore(stratoClient))

)(statsReceiver.scope("SemanticCoreAnnotationStore"))

val underlyingStore = TopicSocialProofStore(

representationScorerStore,

semanticCoreAnnotationStore

)(statsReceiver.scope("TopicSocialProofStore"))

val memcachedStore = ObservedMemcachedReadableStore.fromCacheClient(

backingStore = underlyingStore,

cacheClient = tspUnifiedCacheClient,

ttl = 15.minutes,

asyncUpdate = true

)(

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

statsReceiver = statsReceiver.scope("memCachedTopicSocialProofStore"),

keyToString = { k: TopicSocialProofStore.Query => s"tsps/${k.cacheableQuery}" }

)

val inMemoryCachedStore =

ObservedCachedReadableStore.from[TopicSocialProofStore.Query, Seq[TopicSocialProof]](

memcachedStore,

ttl = 10.minutes,

maxKeys = 16777215, // ~ avg 160B, < 3000MB

cacheName = "topic\_social\_proof\_cache",

windowSize = 10000L

)(statsReceiver.scope("InMemoryCachedTopicSocialProofStore"))

inMemoryCachedStore

}

}