package com.twitter.representation\_manager.store

import com.twitter.finagle.memcached.Client

import com.twitter.finagle.stats.StatsReceiver

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

import com.twitter.representation\_manager.common.MemCacheConfig

import com.twitter.representation\_manager.common.RepresentationManagerDecider

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

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

import com.twitter.simclusters\_v2.stores.SimClustersEmbeddingStore

import com.twitter.simclusters\_v2.summingbird.stores.PersistentTweetEmbeddingStore

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

import com.twitter.simclusters\_v2.thriftscala.EmbeddingType.\_

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

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

import com.twitter.simclusters\_v2.thriftscala.ModelVersion.\_

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

import com.twitter.simclusters\_v2.thriftscala.{SimClustersEmbedding => ThriftSimClustersEmbedding}

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

import com.twitter.storehaus.ReadableStore

import javax.inject.Inject

class TweetSimClustersEmbeddingStore @Inject() (

cacheClient: Client,

globalStats: StatsReceiver,

mhMtlsParams: ManhattanKVClientMtlsParams,

rmsDecider: RepresentationManagerDecider) {

private val stats = globalStats.scope(this.getClass.getSimpleName)

val logFavBasedLongestL2Tweet20M145KUpdatedEmbeddingStore: ReadableStore[

SimClustersEmbeddingId,

SimClustersEmbedding

] = {

val rawStore =

PersistentTweetEmbeddingStore

.longestL2NormTweetEmbeddingStoreManhattan(

mhMtlsParams,

PersistentTweetEmbeddingStore.LogFavBased20m145kUpdatedDataset,

stats

).mapValues(\_.toThrift)

buildMemCacheStore(rawStore, LogFavLongestL2EmbeddingTweet, Model20m145kUpdated)

}

val logFavBasedLongestL2Tweet20M145K2020EmbeddingStore: ReadableStore[

SimClustersEmbeddingId,

SimClustersEmbedding

] = {

val rawStore =

PersistentTweetEmbeddingStore

.longestL2NormTweetEmbeddingStoreManhattan(

mhMtlsParams,

PersistentTweetEmbeddingStore.LogFavBased20m145k2020Dataset,

stats

).mapValues(\_.toThrift)

buildMemCacheStore(rawStore, LogFavLongestL2EmbeddingTweet, Model20m145k2020)

}

val logFavBased20M145KUpdatedTweetEmbeddingStore: ReadableStore[

SimClustersEmbeddingId,

SimClustersEmbedding

] = {

val rawStore =

PersistentTweetEmbeddingStore

.mostRecentTweetEmbeddingStoreManhattan(

mhMtlsParams,

PersistentTweetEmbeddingStore.LogFavBased20m145kUpdatedDataset,

stats

).mapValues(\_.toThrift)

buildMemCacheStore(rawStore, LogFavBasedTweet, Model20m145kUpdated)

}

val logFavBased20M145K2020TweetEmbeddingStore: ReadableStore[

SimClustersEmbeddingId,

SimClustersEmbedding

] = {

val rawStore =

PersistentTweetEmbeddingStore

.mostRecentTweetEmbeddingStoreManhattan(

mhMtlsParams,

PersistentTweetEmbeddingStore.LogFavBased20m145k2020Dataset,

stats

).mapValues(\_.toThrift)

buildMemCacheStore(rawStore, LogFavBasedTweet, Model20m145k2020)

}

private def buildMemCacheStore(

rawStore: ReadableStore[TweetId, ThriftSimClustersEmbedding],

embeddingType: EmbeddingType,

modelVersion: ModelVersion

): ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding] = {

val observedStore: ObservedReadableStore[TweetId, ThriftSimClustersEmbedding] =

ObservedReadableStore(

store = rawStore

)(stats.scope(embeddingType.name).scope(modelVersion.name))

val storeWithKeyMapping = observedStore.composeKeyMapping[SimClustersEmbeddingId] {

case SimClustersEmbeddingId(\_, \_, InternalId.TweetId(tweetId)) =>

tweetId

}

MemCacheConfig.buildMemCacheStoreForSimClustersEmbedding(

storeWithKeyMapping,

cacheClient,

embeddingType,

modelVersion,

stats

)

}

private val underlyingStores: Map[

(EmbeddingType, ModelVersion),

ReadableStore[SimClustersEmbeddingId, SimClustersEmbedding]

] = Map(

// Tweet Embeddings

(LogFavBasedTweet, Model20m145kUpdated) -> logFavBased20M145KUpdatedTweetEmbeddingStore,

(LogFavBasedTweet, Model20m145k2020) -> logFavBased20M145K2020TweetEmbeddingStore,

(

LogFavLongestL2EmbeddingTweet,

Model20m145kUpdated) -> logFavBasedLongestL2Tweet20M145KUpdatedEmbeddingStore,

(

LogFavLongestL2EmbeddingTweet,

Model20m145k2020) -> logFavBasedLongestL2Tweet20M145K2020EmbeddingStore,

)

val tweetSimClustersEmbeddingStore: ReadableStore[

SimClustersEmbeddingId,

SimClustersEmbedding

] = {

SimClustersEmbeddingStore.buildWithDecider(

underlyingStores = underlyingStores,

decider = rmsDecider.decider,

statsReceiver = stats

)

}

}