package com.twitter.simclusters\_v2.summingbird.stores

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

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

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

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

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

import com.twitter.storage.client.manhattan.kv.impl.Component

import com.twitter.storage.client.manhattan.kv.impl.DescriptorP1L0

import com.twitter.storage.client.manhattan.kv.impl.KeyDescriptor

import com.twitter.storage.client.manhattan.kv.impl.ValueDescriptor

import com.twitter.storehaus.ReadableStore

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

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

import com.twitter.storage.client.manhattan.bijections.Bijections.BinaryScalaInjection

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

import com.twitter.conversions.DurationOps.\_

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

import com.twitter.stitch.Stitch

import com.twitter.storage.client.manhattan.bijections.Bijections.LongInjection

import com.twitter.util.Future

/\*\*

\* Manhattan Readable Store to fetch simcluster embedding from a read-write dataset.

\* Only read operations are allowed through this store.

\* @param appId The "application id"

\* @param datasetName The MH dataset name.

\* @param label The human readable label for the finagle thrift client

\* @param mtlsParams Client service identifier to use to authenticate with Manhattan service

\* @param manhattanCluster Manhattan RW cluster

\*\*/

class SimClustersManhattanReadableStoreForReadWriteDataset(

appId: String,

datasetName: String,

label: String,

mtlsParams: ManhattanKVClientMtlsParams,

manhattanCluster: ManhattanCluster = Adama)

extends ReadableStore[SimClustersEmbeddingId, ClustersUserIsInterestedIn] {

/\*

Setting up a new builder to read from Manhattan RW dataset. This is specifically required for

BeT project where we update the MH RW dataset (every 2 hours) using cloud shuttle service.

\*/

val destName = manhattanCluster.wilyName

val endPoint = ManhattanKVEndpointBuilder(ManhattanKVClient(appId, destName, mtlsParams, label))

.defaultGuarantee(Guarantee.SoftDcReadMyWrites)

.build()

val keyDesc = KeyDescriptor(Component(LongInjection), Component()).withDataset(datasetName)

val valueDesc = ValueDescriptor(BinaryScalaInjection(ClustersUserIsInterestedIn))

override def get(

embeddingId: SimClustersEmbeddingId

): Future[Option[ClustersUserIsInterestedIn]] = {

embeddingId match {

case SimClustersEmbeddingId(theEmbeddingType, theModelVersion, InternalId.UserId(userId)) =>

val populatedKey: DescriptorP1L0.FullKey[Long] = keyDesc.withPkey(userId)

// returns result

val mhValue = Stitch.run(endPoint.get(populatedKey, valueDesc))

mhValue.map {

case Some(x) => Option(x.contents)

case \_ => None

}

case \_ => Future.None

}

}

}