package com.twitter.home\_mixer.module

import com.google.inject.Provides

import com.google.inject.name.Named

import com.twitter.bijection.Injection

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.bijection.thrift.ThriftCodec

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.EngagementsReceivedByAuthorCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.RealTimeInteractionGraphUserVertexCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.RealTimeInteractionGraphUserVertexClient

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TimelinesRealTimeAggregateClient

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TopicCountryEngagementCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TopicEngagementCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TweetCountryEngagementCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TweetEngagementCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TwitterListEngagementCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.UserAuthorEngagementCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.UserEngagementCache

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.UserTopicEngagementForNewUserCache

import com.twitter.home\_mixer.util.InjectionTransformerImplicits.\_

import com.twitter.inject.TwitterModule

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.Feature

import com.twitter.ml.{api => ml}

import com.twitter.servo.cache.KeyValueTransformingReadCache

import com.twitter.servo.cache.Memcache

import com.twitter.servo.cache.ReadCache

import com.twitter.servo.util.Transformer

import com.twitter.storehaus\_internal.memcache.MemcacheHelper

import com.twitter.summingbird.batch.Batcher

import com.twitter.summingbird\_internal.bijection.BatchPairImplicits

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.AggregationKey

import com.twitter.timelines.data\_processing.ml\_util.aggregation\_framework.AggregationKeyInjection

import com.twitter.wtf.real\_time\_interaction\_graph.{thriftscala => ig}

import javax.inject.Singleton

object RealtimeAggregateFeatureRepositoryModule

extends TwitterModule

with RealtimeAggregateHelpers {

private val authorIdFeature = new Feature.Discrete("entities.source\_author\_id").getFeatureId

private val countryCodeFeature = new Feature.Text("geo.user\_location.country\_code").getFeatureId

private val listIdFeature = new Feature.Discrete("list.id").getFeatureId

private val userIdFeature = new Feature.Discrete("meta.user\_id").getFeatureId

private val topicIdFeature = new Feature.Discrete("entities.topic\_id").getFeatureId

private val tweetIdFeature = new Feature.Discrete("entities.source\_tweet\_id").getFeatureId

@Provides

@Singleton

@Named(UserTopicEngagementForNewUserCache)

def providesUserTopicEngagementForNewUserCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[(Long, Long), ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD2(userIdFeature, topicIdFeature)

)

}

@Provides

@Singleton

@Named(TwitterListEngagementCache)

def providesTwitterListEngagementCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[Long, ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD1(listIdFeature)

)

}

@Provides

@Singleton

@Named(TopicEngagementCache)

def providesTopicEngagementCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[Long, ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD1(topicIdFeature)

)

}

@Provides

@Singleton

@Named(UserAuthorEngagementCache)

def providesUserAuthorEngagementCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[(Long, Long), ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD2(userIdFeature, authorIdFeature)

)

}

@Provides

@Singleton

@Named(UserEngagementCache)

def providesUserEngagementCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[Long, ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD1(userIdFeature)

)

}

@Provides

@Singleton

@Named(TweetCountryEngagementCache)

def providesTweetCountryEngagementCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[(Long, String), ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD1T1(tweetIdFeature, countryCodeFeature)

)

}

@Provides

@Singleton

@Named(TweetEngagementCache)

def providesTweetEngagementCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[Long, ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD1(tweetIdFeature)

)

}

@Provides

@Singleton

@Named(EngagementsReceivedByAuthorCache)

def providesEngagementsReceivedByAuthorCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[Long, ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD1(authorIdFeature)

)

}

@Provides

@Singleton

@Named(TopicCountryEngagementCache)

def providesTopicCountryEngagementCache(

@Named(TimelinesRealTimeAggregateClient) client: Memcache

): ReadCache[(Long, String), ml.DataRecord] = {

new KeyValueTransformingReadCache(

client,

dataRecordValueTransformer,

keyTransformD1T1(topicIdFeature, countryCodeFeature)

)

}

@Provides

@Singleton

@Named(RealTimeInteractionGraphUserVertexCache)

def providesRealTimeInteractionGraphUserVertexCache(

@Named(RealTimeInteractionGraphUserVertexClient) client: Memcache

): ReadCache[Long, ig.UserVertex] = {

val valueTransformer = BinaryScalaCodec(ig.UserVertex).toByteArrayTransformer()

val underlyingKey: Long => String = {

val cacheKeyPrefix = "user\_vertex"

val defaultBatchID = Batcher.unit.currentBatch

val batchPairInjection = BatchPairImplicits.keyInjection(Injection.connect[Long, Array[Byte]])

MemcacheHelper

.keyEncoder(cacheKeyPrefix)(batchPairInjection)

.compose((k: Long) => (k, defaultBatchID))

}

new KeyValueTransformingReadCache(

client,

valueTransformer,

underlyingKey

)

}

}

trait RealtimeAggregateHelpers {

private def customKeyBuilder[K](prefix: String, f: K => Array[Byte]): K => String = {

// intentionally not implementing injection inverse because it is never used

def g(arr: Array[Byte]) = ???

MemcacheHelper.keyEncoder(prefix)(Injection.build(f)(g))

}

private val keyEncoder: AggregationKey => String = {

val cacheKeyPrefix = ""

val defaultBatchID = Batcher.unit.currentBatch

val batchPairInjection = BatchPairImplicits.keyInjection(AggregationKeyInjection)

customKeyBuilder(cacheKeyPrefix, batchPairInjection)

.compose((k: AggregationKey) => (k, defaultBatchID))

}

protected def keyTransformD1(f1: Long)(key: Long): String = {

val aggregationKey = AggregationKey(Map(f1 -> key), Map.empty)

keyEncoder(aggregationKey)

}

protected def keyTransformD2(f1: Long, f2: Long)(keys: (Long, Long)): String = {

val (k1, k2) = keys

val aggregationKey = AggregationKey(Map(f1 -> k1, f2 -> k2), Map.empty)

keyEncoder(aggregationKey)

}

protected def keyTransformD1T1(f1: Long, f2: Long)(keys: (Long, String)): String = {

val (k1, k2) = keys

val aggregationKey = AggregationKey(Map(f1 -> k1), Map(f2 -> k2))

keyEncoder(aggregationKey)

}

protected val dataRecordValueTransformer: Transformer[DataRecord, Array[Byte]] = ThriftCodec

.toCompact[ml.DataRecord]

.toByteArrayTransformer()

}