package com.twitter.simclusters\_v2.scalding.offline\_job

import com.twitter.algebird.{DecayedValueMonoid, Monoid, OptionMonoid}

import com.twitter.algebird\_internal.thriftscala.{DecayedValue => ThriftDecayedValue}

import com.twitter.scalding.{TypedPipe, \_}

import com.twitter.scalding\_internal.dalv2.DAL

import com.twitter.scalding\_internal.dalv2.remote\_access.{ExplicitLocation, ProcAtla}

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyVal

import com.twitter.simclusters\_v2.common.{Timestamp, TweetId, UserId}

import com.twitter.simclusters\_v2.hdfs\_sources.\_

import com.twitter.simclusters\_v2.summingbird.common.{Configs, ThriftDecayedValueMonoid}

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

import com.twitter.timelineservice.thriftscala.{ContextualizedFavoriteEvent, FavoriteEventUnion}

import java.util.TimeZone

import twadoop\_config.configuration.log\_categories.group.timeline.TimelineServiceFavoritesScalaDataset

object SimClustersOfflineJobUtil {

implicit val timeZone: TimeZone = DateOps.UTC

implicit val dateParser: DateParser = DateParser.default

implicit val modelVersionOrdering: Ordering[PersistedModelVersion] =

Ordering.by(\_.value)

implicit val scoreTypeOrdering: Ordering[PersistedScoreType] =

Ordering.by(\_.value)

implicit val persistedScoresOrdering: Ordering[PersistedScores] = Ordering.by(

\_.score.map(\_.value).getOrElse(0.0)

)

implicit val decayedValueMonoid: DecayedValueMonoid = DecayedValueMonoid(0.0)

implicit val thriftDecayedValueMonoid: ThriftDecayedValueMonoid =

new ThriftDecayedValueMonoid(Configs.HalfLifeInMs)(decayedValueMonoid)

implicit val persistedScoresMonoid: PersistedScoresMonoid =

new PersistedScoresMonoid()(thriftDecayedValueMonoid)

def readInterestedInScalaDataset(

implicit dateRange: DateRange

): TypedPipe[(Long, ClustersUserIsInterestedIn)] = {

//read SimClusters InterestedIn datasets

DAL

.readMostRecentSnapshot(

SimclustersV2InterestedIn20M145KUpdatedScalaDataset,

dateRange.embiggen(Days(30))

)

.withRemoteReadPolicy(ExplicitLocation(ProcAtla))

.toTypedPipe

.map {

case KeyVal(key, value) => (key, value)

}

}

def readTimelineFavoriteData(

implicit dateRange: DateRange

): TypedPipe[(UserId, TweetId, Timestamp)] = {

DAL

.read(TimelineServiceFavoritesScalaDataset, dateRange) // Note: this is a hourly source

.withRemoteReadPolicy(ExplicitLocation(ProcAtla))

.toTypedPipe

.flatMap { cfe: ContextualizedFavoriteEvent =>

cfe.event match {

case FavoriteEventUnion.Favorite(fav) =>

Some((fav.userId, fav.tweetId, fav.eventTimeMs))

case \_ =>

None

}

}

}

class PersistedScoresMonoid(

implicit thriftDecayedValueMonoid: ThriftDecayedValueMonoid)

extends Monoid[PersistedScores] {

private val optionalThriftDecayedValueMonoid =

new OptionMonoid[ThriftDecayedValue]()

override val zero: PersistedScores = PersistedScores()

override def plus(x: PersistedScores, y: PersistedScores): PersistedScores = {

PersistedScores(

optionalThriftDecayedValueMonoid.plus(

x.score,

y.score

)

)

}

def build(value: Double, timeInMs: Double): PersistedScores = {

PersistedScores(Some(thriftDecayedValueMonoid.build(value, timeInMs)))

}

}

}