package com.twitter.interaction\_graph.scio.agg\_flock

import com.spotify.scio.ScioContext

import com.spotify.scio.values.SCollection

import com.twitter.beam.io.dal.DAL

import com.twitter.beam.io.dal.DAL.DiskFormat

import com.twitter.beam.io.dal.DAL.PathLayout

import com.twitter.beam.io.dal.DAL.WriteOptions

import com.twitter.beam.job.ServiceIdentifierOptions

import com.twitter.interaction\_graph.scio.agg\_flock.InteractionGraphAggFlockUtil.\_

import com.twitter.interaction\_graph.scio.common.DateUtil

import com.twitter.interaction\_graph.scio.common.FeatureGeneratorUtil

import com.twitter.interaction\_graph.thriftscala.Edge

import com.twitter.interaction\_graph.thriftscala.FeatureName

import com.twitter.interaction\_graph.thriftscala.Vertex

import com.twitter.scio\_internal.job.ScioBeamJob

import com.twitter.statebird.v2.thriftscala.Environment

import com.twitter.util.Duration

import java.time.Instant

import org.joda.time.Interval

object InteractionGraphAggFlockJob extends ScioBeamJob[InteractionGraphAggFlockOption] {

override protected def configurePipeline(

scioContext: ScioContext,

pipelineOptions: InteractionGraphAggFlockOption

): Unit = {

@transient

implicit lazy val sc: ScioContext = scioContext

implicit lazy val dateInterval: Interval = pipelineOptions.interval

val source = InteractionGraphAggFlockSource(pipelineOptions)

val embiggenInterval = DateUtil.embiggen(dateInterval, Duration.fromDays(7))

val flockFollowsSnapshot = source.readFlockFollowsSnapshot(embiggenInterval)

// the flock snapshot we're reading from has already been filtered for safe/valid users hence no filtering for safeUsers

val flockFollowsFeature =

getFlockFeatures(flockFollowsSnapshot, FeatureName.NumFollows, dateInterval)

val flockMutualFollowsFeature = getMutualFollowFeature(flockFollowsFeature)

val allSCollections = Seq(flockFollowsFeature, flockMutualFollowsFeature)

val allFeatures = SCollection.unionAll(allSCollections)

val (vertex, edges) = FeatureGeneratorUtil.getFeatures(allFeatures)

val dalEnvironment: String = pipelineOptions

.as(classOf[ServiceIdentifierOptions])

.getEnvironment()

val dalWriteEnvironment = if (pipelineOptions.getDALWriteEnvironment != null) {

pipelineOptions.getDALWriteEnvironment

} else {

dalEnvironment

}

vertex.saveAsCustomOutput(

"Write Vertex Records",

DAL.writeSnapshot[Vertex](

InteractionGraphAggFlockVertexSnapshotScalaDataset,

PathLayout.DailyPath(pipelineOptions.getOutputPath + "/aggregated\_flock\_vertex\_daily"),

Instant.ofEpochMilli(dateInterval.getEndMillis),

DiskFormat.Parquet,

Environment.valueOf(dalWriteEnvironment),

writeOption =

WriteOptions(numOfShards = Some((pipelineOptions.getNumberOfShards / 64.0).ceil.toInt))

)

)

edges.saveAsCustomOutput(

"Write Edge Records",

DAL.writeSnapshot[Edge](

InteractionGraphAggFlockEdgeSnapshotScalaDataset,

PathLayout.DailyPath(pipelineOptions.getOutputPath + "/aggregated\_flock\_edge\_daily"),

Instant.ofEpochMilli(dateInterval.getEndMillis),

DiskFormat.Parquet,

Environment.valueOf(dalWriteEnvironment),

writeOption = WriteOptions(numOfShards = Some(pipelineOptions.getNumberOfShards))

)

)

}

}