package com.twitter.interaction\_graph.scio.agg\_client\_event\_logs

import com.spotify.scio.ScioContext

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

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

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

import com.twitter.beam.io.fs.multiformat.PathLayout

import com.twitter.beam.job.ServiceIdentifierOptions

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

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

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

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

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

import org.joda.time.Interval

object InteractionGraphClientEventLogsJob

extends ScioBeamJob[InteractionGraphClientEventLogsOption] {

override protected def configurePipeline(

scioContext: ScioContext,

pipelineOptions: InteractionGraphClientEventLogsOption

): Unit = {

@transient

implicit lazy val sc: ScioContext = scioContext

implicit lazy val jobCounters: InteractionGraphClientEventLogsCountersTrait =

InteractionGraphClientEventLogsCounters

lazy val dateInterval: Interval = pipelineOptions.interval

val sources = InteractionGraphClientEventLogsSource(pipelineOptions)

val userInteractions = sources.readUserInteractions(dateInterval)

val rawUsers = sources.readCombinedUsers()

val safeUsers = UserUtil.getValidUsers(rawUsers)

val (vertex, edges) = InteractionGraphClientEventLogsUtil.process(userInteractions, safeUsers)

val dalEnvironment: String = pipelineOptions

.as(classOf[ServiceIdentifierOptions])

.getEnvironment()

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

pipelineOptions.getDALWriteEnvironment

} else {

dalEnvironment

}

vertex.saveAsCustomOutput(

"Write Vertex Records",

DAL.write[Vertex](

InteractionGraphAggClientEventLogsVertexDailyScalaDataset,

PathLayout.DailyPath(

pipelineOptions.getOutputPath + "/aggregated\_client\_event\_logs\_vertex\_daily"),

dateInterval,

DiskFormat.Parquet,

Environment.valueOf(dalWriteEnvironment),

writeOption =

WriteOptions(numOfShards = Some((pipelineOptions.getNumberOfShards / 32.0).ceil.toInt))

)

)

edges.saveAsCustomOutput(

"Write Edge Records",

DAL.write[Edge](

InteractionGraphAggClientEventLogsEdgeDailyScalaDataset,

PathLayout.DailyPath(

pipelineOptions.getOutputPath + "/aggregated\_client\_event\_logs\_edge\_daily"),

dateInterval,

DiskFormat.Parquet,

Environment.valueOf(dalWriteEnvironment),

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

)

)

}

}