package com.twitter.graph\_feature\_service.common

import com.twitter.conversions.DurationOps.\_

import com.twitter.util.Duration

import com.twitter.util.Time

import java.nio.ByteBuffer

import scala.util.hashing.MurmurHash3

object Configs {

// NOTE: notify #recos-platform slack room, if you want to change this.

// This SHOULD be updated together with NUM\_SHARDS in worker.aurora

final val NumGraphShards: Int = 40

final val TopKRealGraph: Int = 512

final val BaseHdfsPath: String = "/user/cassowary/processed/gfs/constant\_db/"

// whether or not to write in\_value and out\_value graphs. Used in the scalding job.

final val EnableValueGraphs: Boolean = true

// whether or not to write in\_key and out\_key graphs. Used in the scalding job.

final val EnableKeyGraphs: Boolean = false

final val FollowOutValPath: String = "follow\_out\_val/"

final val FollowOutKeyPath: String = "follow\_out\_key/"

final val FollowInValPath: String = "follow\_in\_val/"

final val FollowInKeyPath: String = "follow\_in\_key/"

final val MutualFollowValPath: String = "mutual\_follow\_val/"

final val MutualFollowKeyPath: String = "mutual\_follow\_key/"

final val FavoriteOutValPath: String = "favorite\_out\_val/"

final val FavoriteInValPath: String = "favorite\_in\_val/"

final val FavoriteOutKeyPath: String = "favorite\_out\_key/"

final val FavoriteInKeyPath: String = "favorite\_in\_key/"

final val RetweetOutValPath: String = "retweet\_out\_val/"

final val RetweetInValPath: String = "retweet\_in\_val/"

final val RetweetOutKeyPath: String = "retweet\_out\_key/"

final val RetweetInKeyPath: String = "retweet\_in\_key/"

final val MentionOutValPath: String = "mention\_out\_val/"

final val MentionInValPath: String = "mention\_in\_val/"

final val MentionOutKeyPath: String = "mention\_out\_key/"

final val MentionInKeyPath: String = "mention\_in\_key/"

final val MemCacheTTL: Duration = 8.hours

final val RandomSeed: Int = 39582942

def getTimedHdfsShardPath(shardId: Int, path: String, time: Time): String = {

val timeStr = time.format("yyyy/MM/dd")

s"$path/$timeStr/shard\_$shardId"

}

def getHdfsPath(path: String, overrideBaseHdfsPath: Option[String] = None): String = {

val basePath = overrideBaseHdfsPath.getOrElse(BaseHdfsPath)

s"$basePath$path"

}

private def hash(kArr: Array[Byte], seed: Int): Int = {

MurmurHash3.bytesHash(kArr, seed) & 0x7fffffff // keep positive

}

private def hashLong(l: Long, seed: Int): Int = {

hash(ByteBuffer.allocate(8).putLong(l).array(), seed)

}

def shardForUser(userId: Long): Int = {

hashLong(userId, RandomSeed) % NumGraphShards

}

}