package com.twitter.servo.gate

import com.google.common.annotations.VisibleForTesting

import com.google.common.util.concurrent.RateLimiter

import com.twitter.servo.util

import java.util.concurrent.TimeUnit

/\*\*

\* A Rate Limiting Gate backed by com.google.common.util.concurrent.RateLimiter

\* http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/util/concurrent/RateLimiter.html

\*/

object RateLimitingGate {

/\*\*

\* Creates a Gate[Int] that returns true if acquiring <gate\_input> number of permits

\* from the ratelimiter succeeds.

\*/

def weighted(permitsPerSecond: Double): util.Gate[Int] = {

val rateLimiter: RateLimiter = RateLimiter.create(permitsPerSecond)

util.Gate { rateLimiter.tryAcquire(\_, 0, TimeUnit.SECONDS) }

}

/\*\*

\* Creates a Gate[Unit] that returns true if acquiring a permit from the ratelimiter succeeds.

\*/

def uniform(permitsPerSecond: Double): util.Gate[Unit] = {

weighted(permitsPerSecond) contramap { \_ =>

1

}

}

/\*\*

\* Creates a Gate[Unit] with floating limit. Could be used with deciders.

\*/

def dynamic(permitsPerSecond: => Double): util.Gate[Unit] =

dynamic(RateLimiter.create, permitsPerSecond)

@VisibleForTesting

def dynamic(

rateLimiterFactory: Double => RateLimiter,

permitsPerSecond: => Double

): util.Gate[Unit] = {

val rateLimiter: RateLimiter = rateLimiterFactory(permitsPerSecond)

util.Gate { \_ =>

val currentRate = permitsPerSecond

if (rateLimiter.getRate != currentRate) {

rateLimiter.setRate(currentRate)

}

rateLimiter.tryAcquire(0L, TimeUnit.SECONDS)

}

}

}

@deprecated("Use RateLimitingGate.uniform", "2.8.2")

class RateLimitingGate[T](permitsPerSecond: Double) extends util.Gate[T] {

private[this] val rateLimiter: RateLimiter = RateLimiter.create(permitsPerSecond)

/\*\*

\* If a "permit" is available, this method acquires it and returns true

\* Else returns false immediately without waiting

\*/

override def apply[U](u: U)(implicit asT: <:<[U, T]): Boolean =

rateLimiter.tryAcquire(1, 0, TimeUnit.SECONDS)

}