package com.twitter.servo.util

import com.twitter.logging.{Level, Logger}

import com.twitter.util.{Duration, Time}

import com.twitter.conversions.DurationOps.\_

import java.util.concurrent.atomic.AtomicLong

object RateLimitingLogger {

private[util] val DefaultLoggerName = "servo"

private[util] val DefaultLogInterval = 500.milliseconds

}

/\*\*

\* Class that makes it easier to rate-limit log messages, either by call site, or by

\* logical grouping of messages.

\* @param interval the interval in which messages should be rate limited

\* @param logger the logger to use

\*/

class RateLimitingLogger(

interval: Duration = RateLimitingLogger.DefaultLogInterval,

logger: Logger = Logger(RateLimitingLogger.DefaultLoggerName)) {

private[this] val last: AtomicLong = new AtomicLong(0L)

private[this] val sinceLast: AtomicLong = new AtomicLong(0L)

private[this] val intervalNanos = interval.inNanoseconds

private[this] val intervalMsString = interval.inMilliseconds.toString

private[this] def limited(action: Long => Unit): Unit = {

val now = Time.now.inNanoseconds

val lastNanos = last.get()

if (now - lastNanos > intervalNanos) {

if (last.compareAndSet(lastNanos, now)) {

val currentSinceLast = sinceLast.getAndSet(0L)

action(currentSinceLast)

}

} else {

sinceLast.incrementAndGet()

}

}

def log(msg: => String, level: Level = Level.ERROR): Unit = {

limited { currentSinceLast: Long =>

logger(

level,

"%s (group is logged at most once every %s ms%s)".format(

msg,

intervalMsString,

if (currentSinceLast > 0) {

s", ${currentSinceLast} occurrences since last"

} else ""

)

)

}

}

def logThrowable(t: Throwable, msg: => String, level: Level = Level.ERROR): Unit = {

limited { currentSinceLast: Long =>

logger(

level,

t,

"%s (group is logged at most once every %s ms%s)".format(

msg,

intervalMsString,

if (currentSinceLast > 0) {

s", ${currentSinceLast} occurrences since last"

} else ""

)

)

}

}

}