package com.twitter.ann.service.query\_server.common.throttling

import com.twitter.util.Duration

trait ThrottlingInstrument {

def sample(): Unit

def percentageOfTimeSpentThrottling(): Double

def disabled: Boolean

}

class WindowedThrottlingInstrument(

stepFrequency: Duration,

windowLengthInFrequencySteps: Int,

reader: AuroraCPUStatsReader)

extends ThrottlingInstrument {

private[this] val throttlingChangeHistory: WindowedStats = new WindowedStats(

windowLengthInFrequencySteps)

private[this] val cpuQuota: Double = reader.cpuQuota

// The total number of allotted CPU time per step (in nanos).

private[this] val assignedCpu: Duration = stepFrequency \* cpuQuota

private[this] val assignedCpuNs: Long = assignedCpu.inNanoseconds

@volatile private[this] var previousThrottledTimeNs: Long = 0

/\*\*

\* If there isn't a limit on how much cpu the container can use, aurora

\* throttling will never kick in.

\*/

final def disabled: Boolean = cpuQuota <= 0

def sample(): Unit = sampleThrottling() match {

case Some(load) =>

throttlingChangeHistory.add(load)

case None => ()

}

private[this] def sampleThrottling(): Option[Long] = reader.throttledTimeNanos().map {

throttledTimeNs =>

val throttlingChange = throttledTimeNs - previousThrottledTimeNs

previousThrottledTimeNs = throttledTimeNs

throttlingChange

}

// Time spent throttling over windowLength, normalized by number of CPUs

def percentageOfTimeSpentThrottling(): Double = {

math.min(1, throttlingChangeHistory.sum.toDouble / assignedCpuNs)

}

}