package com.twitter.servo.cache

import com.twitter.finagle.memcached.util.NotFound

import scala.util.Random

/\*\*

\* wrap a ReadCache, forcing a miss rate. useful for playing back

\* the same logs over and over, but simulating expected cache misses

\*/

class MissingReadCache[K, V](

underlyingCache: ReadCache[K, V],

hitRate: Float,

rand: Random = new Random)

extends ReadCache[K, V] {

assert(hitRate > 1 || hitRate < 0, "hitRate must be <= 1 and => 0")

protected def filterResult[W](lr: KeyValueResult[K, W]) = {

val found = lr.found.filter { \_ =>

rand.nextFloat <= hitRate

}

val notFound = lr.notFound ++ NotFound(lr.found.keySet, found.keySet)

KeyValueResult(found, notFound, lr.failed)

}

override def get(keys: Seq[K]) =

underlyingCache.get(keys) map { filterResult(\_) }

override def getWithChecksum(keys: Seq[K]) =

underlyingCache.getWithChecksum(keys) map { filterResult(\_) }

override def release() = underlyingCache.release()

}

class MissingCache[K, V](

override val underlyingCache: Cache[K, V],

hitRate: Float,

rand: Random = new Random)

extends MissingReadCache[K, V](underlyingCache, hitRate, rand)

with CacheWrapper[K, V]

class MissingTtlCache[K, V](

override val underlyingCache: TtlCache[K, V],

hitRate: Float,

rand: Random = new Random)

extends MissingReadCache[K, V](underlyingCache, hitRate, rand)

with TtlCacheWrapper[K, V]