package com.twitter.servo.repository

import com.twitter.servo.cache.\_

import com.twitter.util.Future

class CachingCounterKeyValueRepository[K](

underlying: CounterKeyValueRepository[K],

cache: CounterCache[K],

observer: CacheObserver = NullCacheObserver)

extends CounterKeyValueRepository[K] {

def apply(keys: Seq[K]): Future[KeyValueResult[K, Long]] = {

val uniqueKeys = keys.distinct

cache.get(uniqueKeys) flatMap { cachedResults =>

recordResults(cachedResults)

val missed = cachedResults.notFound ++ cachedResults.failed.keySet

readThrough(missed.toSeq) map { readResults =>

KeyValueResult(cachedResults.found) ++ readResults

}

}

}

private def readThrough(keys: Seq[K]): Future[KeyValueResult[K, Long]] =

if (keys.isEmpty) {

KeyValueResult.emptyFuture

} else {

underlying(keys) onSuccess { readResults =>

for ((k, v) <- readResults.found) {

cache.add(k, v)

}

}

}

private def recordResults(cachedResults: KeyValueResult[K, Long]): Unit = {

cachedResults.found.keys foreach { key =>

observer.hit(key.toString)

}

cachedResults.notFound foreach { key =>

observer.miss(key.toString)

}

observer.failure(cachedResults.failed.size)

}

}