package com.twitter.simclusters\_v2.summingbird.common

import com.twitter.algebird.DecayedValue

import com.twitter.algebird.DecayedValueMonoid

import com.twitter.algebird.Monoid

import com.twitter.algebird\_internal.injection.DecayedValueImplicits

import com.twitter.algebird\_internal.thriftscala.{DecayedValue => ThriftDecayedValue}

/\*\*

\* Monoid for ThriftDecayedValue

\*/

class ThriftDecayedValueMonoid(halfLifeInMs: Long)(implicit decayedValueMonoid: DecayedValueMonoid)

extends Monoid[ThriftDecayedValue] {

override val zero: ThriftDecayedValue = DecayedValueImplicits.toThrift(decayedValueMonoid.zero)

override def plus(x: ThriftDecayedValue, y: ThriftDecayedValue): ThriftDecayedValue = {

DecayedValueImplicits.toThrift(

decayedValueMonoid

.plus(DecayedValueImplicits.toThrift.invert(x), DecayedValueImplicits.toThrift.invert(y))

)

}

def build(value: Double, timeInMs: Double): ThriftDecayedValue = {

DecayedValueImplicits.toThrift(

DecayedValue.build(value, timeInMs, halfLifeInMs)

)

}

/\*\*

\* decay to a timestamp; note that timestamp should be in Ms, and do not use scaledTime!

\*/

def decayToTimestamp(

thriftDecayedValue: ThriftDecayedValue,

timestampInMs: Double

): ThriftDecayedValue = {

this.plus(thriftDecayedValue, this.build(0.0, timestampInMs))

}

}

object ThriftDecayedValueMonoid {

// add the implicit class so that a decayed value can direct call .plus, .decayedValueOfTime and

// so on.

implicit class EnrichedThriftDecayedValue(

thriftDecayedValue: ThriftDecayedValue

)(

implicit thriftDecayedValueMonoid: ThriftDecayedValueMonoid) {

def plus(other: ThriftDecayedValue): ThriftDecayedValue = {

thriftDecayedValueMonoid.plus(thriftDecayedValue, other)

}

// decay to a timestamp; note that timestamp should be in Ms

def decayToTimestamp(timeInMs: Double): ThriftDecayedValue = {

thriftDecayedValueMonoid.decayToTimestamp(thriftDecayedValue, timeInMs)

}

}

}