package com.twitter.ann.serialization

import com.twitter.scrooge.{ThriftStruct, ThriftStructCodec}

import java.io.{InputStream, OutputStream}

import org.apache.thrift.protocol.TBinaryProtocol

import org.apache.thrift.transport.{TIOStreamTransport, TTransportException}

/\*\*

\* Class that can serialize and deserialize an iterator of thrift objects.

\* This class can do things lazily so there is no need to have all the object into memory.

\*/

class ThriftIteratorIO[T <: ThriftStruct](

codec: ThriftStructCodec[T]) {

def toOutputStream(

iterator: Iterator[T],

outputStream: OutputStream

): Unit = {

val protocol = (new TBinaryProtocol.Factory).getProtocol(new TIOStreamTransport(outputStream))

iterator.foreach { thriftObject =>

codec.encode(thriftObject, protocol)

}

}

/\*\*

\* Returns an iterator that lazily reads from an inputStream.

\* @return

\*/

def fromInputStream(

inputStream: InputStream

): Iterator[T] = {

ThriftIteratorIO.getIterator(codec, inputStream)

}

}

object ThriftIteratorIO {

private def getIterator[T <: ThriftStruct](

codec: ThriftStructCodec[T],

inputStream: InputStream

): Iterator[T] = {

val protocol = (new TBinaryProtocol.Factory).getProtocol(new TIOStreamTransport(inputStream))

def getNext: Option[T] =

try {

Some(codec.decode(protocol))

} catch {

case e: TTransportException if e.getType == TTransportException.END\_OF\_FILE =>

inputStream.close()

None

}

Iterator

.continually[Option[T]](getNext)

.takeWhile(\_.isDefined)

// It should be safe to call get on here since we are only take the defined ones.

.map(\_.get)

}

}