最近一段时间在做项目的过程中，遇到一个非常棘手的问题，我程序代码是用Python写的，主要是逐日逐时计算太阳能辐射值，计算需要非常长的时间，但是我们一开始写出来的程序代码经常会跑一段时间因为CPU内存消耗殆尽而死掉，也就是常说的内存溢出，为了找到为什么会存在内存溢出的问题，我开始以为是Python的对象引用机制和垃圾回收机制的原因，看了一下这块的知识，顺便在这里做个总结。Python中一般我们不需要去管理内存，他的内存机制是根据某个对象的引用次数来决定是否垃圾回收释放对象所占得内存空间，当我们创建一个变量并赋值时，是现在内存上申请一块空间，然后将我们的值存在放该内存空间，而我们的变量是对这个值的引用，也就是说当对象被创建并赋值给变量时，该对象的引用计数就+1,那么什么情况下对象的引用次数会增加呢：

对象被创建

X=3.14

或者另外的别名被创建

Y=X

作为参数传递给函数

Foobar(X)

作为容器对象的一个元素

MYLIST=[,,,X,]

为了销毁对象释放内存空间，需要对象的引用计算减少。最明显的例子就是当引用离开其作用范围时，这种情况是最经常出现在函数运行结束，所有在函数体内的局部变量都会被自动销毁，对象的引用计数也会随之减少，具体有以下几种情况：

一个本地引用离开了其作用范围，比例Foorbar(x)函数运行结束时

对象的别名被显式销毁

Del Y

对象的别名赋值给其他对象

X=2.16

对象被从一个窗口对象中移除或者删除窗口

MYLIST.remove(X) del MYLIST

\*\*\*\*\*注意，Python中的del语句不是删除对象释放内存，而是将变量所赋值的对象引用次数减1,如果该对象的引用次数为0，则该对象成为垃圾回收机制的回收对象。

下面的代码可打印出代码里某些变量的引用计数

Import sys

print sys.getrefcount(a)

我发现在我的代码里确实有些对象在代码结束后引用计数竟然没有减少，在我的代码里有这样一部分

arcpy.AddMessage("Saving Data...")

OutputPath = workSpacePath + "\\Result\_CLoud2016\_test\\"+str(region)+"\\"

#Get IpRaster Path

outIpPath = OutputPath + RightTempDatetime

if os.path.exists(outIpPath):

MyUtility.delete\_file\_folder(outIpPath)

os.makedirs(outIpPath)

outIpRaster = outIpPath + "\\" + Inputrasterlen + "\_IpC" + RightTempDatetime

SRTran\_Arc.Numpy2Raster(tempIp,outIpRaster,dsm\_Xmin,dsm\_Ymin,XCellSize,YCellSize)

arcpy.AddMessage("Saving Ip Finish...")

其中SRTran\_Arc.Numpy2Raster封装的函数如下所示，就是arcpy里面将Numpy转换成栅格数据

def Numpy2Raster(myArray,path,Xmin,Ymin,Xcellsize,Ycellsize):

point = arcpy.Point(Xmin, Ymin)

myRaster = arcpy.NumPyArrayToRaster(myArray,point,Xcellsize,Ycellsize)

myRaster.save(path)

因此我怀疑是由于arcpy模块里封装的函数的原因导致内存没释放，其实在这里我也发现一个问题，就是我查看每个对象的引用次数时，发现程序结束后对象的引用计数虽然有减少，但并不为0，我不知道对象到底有没有被释放，因此我只能采用别的方法，通过更直接的内存检测方式来看看哪一个环节出问题了

Python中有一个Memory\_profile模块，可以通过pip的方式安装

pip install memory\_profiler

通过这个模块，我们可以分析代码每行所消耗的系统内存，安装完该模块后，将需要分析的代码封装成一个函数，然后在函数前加修饰符@profile,代码运行结束即可逐行检查每行代码所占内存

def func():

\*\*\*\*\*

If \_\_name\_\_=”\_\_main\_\_”:

def func()

运行结果会打印四列如下所示

Line # Mem usage Increment Line Contents

================================================

行数 内存消耗 内存增加 行代码

通过这种方式可以检查到哪一环节内存消耗增加了，增加了多少，以及是否有释放内存

在我的代码里存在一个for循环来计算一年中每一天的天阳能辐射值，计算结果为

tempIp

tempId

tempIdu

tempIr

tempIt

然后arcpy.NumPyArrayToRaster将计算结果存储为raster,为了简化代码，我将代码中一些无关紧要的部分删除再贴上来，下面是我循环计算第一天的内存检测，认真看一下，126行代码为for循环之前系统消耗内存818.4 MiB，我们先不看第131行代码，然后继续看133到138行代码，可以看到由于创建tempIp、tempId、tempIdu、tempIr、tempIt导致内存的上升，每个对象的创建导致内存增加181.9 MiB，一共增加 909.5 MiB，然后一直到代码执行结束，都没有释放这一部分的内存，导致我们到下一循环开始时系统内存消耗增长为2428.0 MiB，具体看131行。

Line # Mem usage Increment Line Contents

================================================

21 221.6 MiB 0.0 MiB @profile

22 def my\_func():

126 818.4 MiB 0.0 MiB totalCount = rowCount \* columnCount

131 2428.0 MiB 1609.6 MiB for i in xrange(0,int(deltaDays) + 1):

133 1000.4 MiB 181.9 MiB tempIp = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

134 1182.3 MiB 181.9 MiB tempId = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

135 1364.2 MiB 181.9 MiB tempIdu = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

136 1546.1 MiB 181.9 MiB tempIr = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

137 1728.0 MiB 181.9 MiB tempIt = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32)

138 1728.0 MiB 0.0 MiB print 'tempIp refcount', sys.getrefcount(tempIp)

250 2426.6 MiB 0.0 MiB tempIp = tempIp.reshape(rowCount,columnCount)

251 2426.6 MiB 0.0 MiB tempId = tempId.reshape(rowCount,columnCount)

252 2426.6 MiB 0.0 MiB tempIr = tempIr.reshape(rowCount,columnCount)

253 2426.6 MiB 0.0 MiB tempIdu= tempIdu.reshape(rowCount,columnCount)

254 2426.6 MiB 0.0 MiB tempIt = SRTran.GetIt(tempIp,tempId,tempIr)

255 2426.6 MiB 0.0 MiB tempIp = tempIp.astype(numpy.int32)

256 2426.6 MiB 0.0 MiB tempId = tempId.astype(numpy.int32)

257 2426.6 MiB 0.0 MiB tempIr= tempIr.astype(numpy.int32)

258 2426.6 MiB 0.0 MiB tempIdu = tempIdu.astype(numpy.int32)

259 2426.6 MiB 0.0 MiB tempIt = tempIt.astype(numpy.int32)

262 #print tempIt

263 2426.6 MiB 0.0 MiB arcpy.AddMessage("Saving Data...")

264 2426.6 MiB 0.0 MiB OutputPath = workSpacePath + "\\Result\_CLoud2016\_test\\"+str(region)+"\\"

265 #Get IpRaster Path

266 2426.6 MiB 0.0 MiB outIpPath = OutputPath + RightTempDatetime

267 2426.6 MiB 0.0 MiB if os.path.exists(outIpPath):

268 MyUtility.delete\_file\_folder(outIpPath)

269 2426.6 MiB 0.0 MiB os.makedirs(outIpPath)

270 2426.6 MiB 0.0 MiB outIpRaster = outIpPath + "\\" + Inputrasterlen + "\_IpC" + RightTempDatetime

271 2426.6 MiB 0.0 MiB point = arcpy.Point(dsm\_Xmin, dsm\_Ymin)

272 2614.2 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempIp, point,XCellSize,YCellSize)

273 2427.5 MiB -186.7 MiB myRaster.save(outIpRaster)

274 2427.5 MiB 0.0 MiB del myRaster

275 2427.5 MiB 0.0 MiB arcpy.AddMessage("Saving Ip Finish...")

#Get IdRaster Path

280 2427.5 MiB 0.0 MiB outIdPath = OutputPath + RightTempDatetime

281 #if os.path.exists(outIdPath):

282 #MyUtility.delete\_file\_folder(outIdPath)

283 #os.makedirs(outIdPath)

284 2427.5 MiB 0.0 MiB outIdRaster = outIdPath + "\\" + Inputrasterlen + "\_IdC" + RightTempDatetime

285 2615.1 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempId, point,XCellSize,YCellSize)

286 2427.5 MiB -187.6 MiB myRaster.save(outIdRaster)

287 2427.5 MiB 0.0 MiB del myRaster

288 2427.5 MiB 0.0 MiB arcpy.AddMessage("Saving Id Finish...")

289

291 #Get IrRaster Path

292 2427.5 MiB 0.0 MiB outIrPath = OutputPath + RightTempDatetime

293 #if os.path.exists(outIrPath):

294 #MyUtility.delete\_file\_folder(outIrPath)

295 #os.makedirs(outIrPath)

296 2427.5 MiB 0.0 MiB outIrRaster = outIrPath + "\\" + Inputrasterlen + "\_IrC" + RightTempDatetime

297 2615.1 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempIr, point,XCellSize,YCellSize)

298 2427.6 MiB -187.5 MiB myRaster.save(outIrRaster)

299 2427.6 MiB 0.0 MiB del myRaster

300 2427.6 MiB 0.0 MiB arcpy.AddMessage("Saving Ir Finish...")

301

302

303 #Get ItRaster Path

304 2427.6 MiB 0.0 MiB outItPath = OutputPath + RightTempDatetime

305 #if os.path.exists(outItPath):

306 #MyUtility.delete\_file\_folder(outItPath)

307 #os.makedirs(outItPath)

308 2427.6 MiB 0.0 MiB outItRaster = outItPath + "\\" + Inputrasterlen + "\_ItC" + RightTempDatetime

309 2615.2 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempIt, point,XCellSize,YCellSize)

310 2427.9 MiB -187.3 MiB myRaster.save(outItRaster)

311 2427.9 MiB 0.0 MiB del myRaster

312 2427.9 MiB 0.0 MiB arcpy.AddMessage("Saving It Finish...")

313

314 #Get IdurationRaster Path

315 2427.9 MiB 0.0 MiB outIduPath = OutputPath + RightTempDatetime

316 #if os.path.exists(outIduPath):

317 #MyUtility.delete\_file\_folder(outIduPath)

318 #os.makedirs(outIduPath)

319 2427.9 MiB 0.0 MiB outIduRaster = outIduPath + "\\" + Inputrasterlen + "\_Idu" + RightTempDatetime

320 2615.5 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempIdu, point,XCellSize,YCellSize)

321 2615.5 MiB 0.0 MiB del tempIdu

322 2427.9 MiB -187.6 MiB myRaster.save(outIduRaster)

323 2427.9 MiB 0.0 MiB del myRaster

324 2427.9 MiB 0.0 MiB arcpy.AddMessage("Saving Idu Finish...")

325 2428.0 MiB 0.1 MiB print gc.get\_referrers(tempIp)

326 2428.0 MiB 0.0 MiB dayend=time.clock()

327 2428.0 MiB 0.0 MiB print dayend-daystart

好的，第一个循环结束了，为了让代码继续执行，我们将循环天数设置为2天，这样可以观察代码继续执行第二个循环后系统内存的消耗情况。接下来我将内存检测的代码复制上来，可以看到第一天结束时代码运行到132时系统消耗内存2373.1 MiB(因为我重新运行了代码，所以这里与上面的2428.0 MiB有点差异)，继续创建tempIp、tempId、tempIdu、tempIr、tempIt，可以看到系统内存消耗又上升到3282.7 MiB，增长了909.5 MiB，一直到代码该循环结束，这一部分的内存仍没释放，开始第三天循环时，系统内存上升到3338.2 MiB

Line # Mem usage Increment Line Contents

================================================

21 222.1 MiB 0.0 MiB @profile

22 def my\_func():

129

130

131 3338.2 MiB 2513.4 MiB for i in xrange(0,int(deltaDays) + 1):

132 2373.1 MiB -965.1 MiB daystart=time.clock()

133 2555.0 MiB 181.9 MiB tempIp = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

134 2736.9 MiB 181.9 MiB tempId = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

135 2918.8 MiB 181.9 MiB tempIdu = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

136 3100.8 MiB 181.9 MiB tempIr = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32).reshape(-1)

137 3282.7 MiB 181.9 MiB tempIt = numpy.zeros\_like(slopeNumpy,dtype=numpy.float32)

138 3282.7 MiB 0.0 MiB print 'tempIp refcount', sys.getrefcount(tempIp)

139

250 3337.3 MiB 0.0 MiB tempIp = tempIp.reshape(rowCount,columnCount)

251 3337.3 MiB 0.0 MiB tempId = tempId.reshape(rowCount,columnCount)

252 3337.3 MiB 0.0 MiB tempIr = tempIr.reshape(rowCount,columnCount)

253 3337.3 MiB 0.0 MiB tempIdu= tempIdu.reshape(rowCount,columnCount)

254 3337.3 MiB 0.0 MiB tempIt = SRTran.GetIt(tempIp,tempId,tempIr)

255 3337.3 MiB 0.0 MiB tempIp = tempIp.astype(numpy.int32)

256 3337.3 MiB 0.0 MiB tempId = tempId.astype(numpy.int32)

257 3337.3 MiB 0.0 MiB tempIr= tempIr.astype(numpy.int32)

258 3337.3 MiB 0.0 MiB tempIdu = tempIdu.astype(numpy.int32)

259 3337.3 MiB 0.0 MiB tempIt = tempIt.astype(numpy.int32)

260 #print arcpy.CheckOutExtension("Spatial")

261

262 #print tempIt

263 3337.3 MiB 0.0 MiB arcpy.AddMessage("Saving Data...")

264 3337.3 MiB 0.0 MiB OutputPath = workSpacePath + "\\Result\_CLoud2016\_test\\"+str(region)+"\\"

265 #Get IpRaster Path

266 3337.3 MiB 0.0 MiB outIpPath = OutputPath + RightTempDatetime

267 3337.3 MiB 0.0 MiB if os.path.exists(outIpPath):

268 MyUtility.delete\_file\_folder(outIpPath)

269 3337.3 MiB 0.0 MiB os.makedirs(outIpPath)

270 3337.3 MiB 0.0 MiB outIpRaster = outIpPath + "\\" + Inputrasterlen + "\_IpC" + RightTempDatetime

271 3337.3 MiB 0.0 MiB point = arcpy.Point(dsm\_Xmin, dsm\_Ymin)

272 3524.9 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempIp, point,XCellSize,YCellSize)

273 3338.2 MiB -186.7 MiB myRaster.save(outIpRaster)

274 3338.2 MiB 0.0 MiB del myRaster

275 3338.2 MiB 0.0 MiB arcpy.AddMessage("Saving Ip Finish...")

276

277

278

279 #Get IdRaster Path

280 3338.2 MiB 0.0 MiB outIdPath = OutputPath + RightTempDatetime

281 #if os.path.exists(outIdPath):

282 #MyUtility.delete\_file\_folder(outIdPath)

283 #os.makedirs(outIdPath)

284 3338.2 MiB 0.0 MiB outIdRaster = outIdPath + "\\" + Inputrasterlen + "\_IdC" + RightTempDatetime

285 3525.8 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempId, point,XCellSize,YCellSize)

286 3338.2 MiB -187.6 MiB myRaster.save(outIdRaster)

287 3338.2 MiB 0.0 MiB del myRaster

288 3338.2 MiB 0.0 MiB arcpy.AddMessage("Saving Id Finish...")

289

290

291 #Get IrRaster Path

292 3338.2 MiB 0.0 MiB outIrPath = OutputPath + RightTempDatetime

293 #if os.path.exists(outIrPath):

294 #MyUtility.delete\_file\_folder(outIrPath)

295 #os.makedirs(outIrPath)

296 3338.2 MiB 0.0 MiB outIrRaster = outIrPath + "\\" + Inputrasterlen + "\_IrC" + RightTempDatetime

297 3525.4 MiB 187.1 MiB myRaster = arcpy.NumPyArrayToRaster(tempIr, point,XCellSize,YCellSize)

298 3337.8 MiB -187.5 MiB myRaster.save(outIrRaster)

299 3337.8 MiB 0.0 MiB del myRaster

300 3337.8 MiB 0.0 MiB arcpy.AddMessage("Saving Ir Finish...")

301

302

303 #Get ItRaster Path

304 3337.8 MiB 0.0 MiB outItPath = OutputPath + RightTempDatetime

305 #if os.path.exists(outItPath):

306 #MyUtility.delete\_file\_folder(outItPath)

307 #os.makedirs(outItPath)

308 3337.8 MiB 0.0 MiB outItRaster = outItPath + "\\" + Inputrasterlen + "\_ItC" + RightTempDatetime

309 3525.4 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempIt, point,XCellSize,YCellSize)

310 3338.2 MiB -187.2 MiB myRaster.save(outItRaster)

311 3338.2 MiB 0.0 MiB del myRaster

312 3338.2 MiB 0.0 MiB arcpy.AddMessage("Saving It Finish...")

313

314 #Get IdurationRaster Path

315 3338.2 MiB 0.0 MiB outIduPath = OutputPath + RightTempDatetime

316 #if os.path.exists(outIduPath):

317 #MyUtility.delete\_file\_folder(outIduPath)

318 #os.makedirs(outIduPath)

319 3338.2 MiB 0.0 MiB outIduRaster = outIduPath + "\\" + Inputrasterlen + "\_Idu" + RightTempDatetime

320 3525.8 MiB 187.6 MiB myRaster = arcpy.NumPyArrayToRaster(tempIdu, point,XCellSize,YCellSize)

321 3525.8 MiB 0.0 MiB del tempIdu

322 3338.2 MiB -187.6 MiB myRaster.save(outIduRaster)

323 3338.2 MiB 0.0 MiB del myRaster

324 3338.2 MiB 0.0 MiB arcpy.AddMessage("Saving Idu Finish...")

325 3338.2 MiB 0.0 MiB print gc.get\_referrers(tempIp)

326 3338.2 MiB 0.0 MiB dayend=time.clock()

327 3338.2 MiB 0.0 MiB print dayend-daystart

从上面的分析可以看出，每一个循环计算出来的tempIp、tempId、tempIdu、tempIr、tempIt对象在系统内存里没有被释放，导致随着代码循环的增加，系统内存消耗逐渐增加，最终导致内存溢出，程序死掉，然后我尝试在这些变量被存成raster之后将这几个变量显式del了，但是还是没有释放该内存，再结合前面我分析的对象引用次数，我怀疑由于arcpy.NumPyArrayToRaster这个函数在调用tempIp、tempId、tempIdu、tempIr、tempIt这几个变量后没有将对象引用次数减1，导致垃圾回收机制无法回收这几个对象，因此我就想着有没有其他方法可以将Numpy读成Raster存储下来，之前用过GDAL，我就修改了我的代码,至于GDAL如何将Numpy 读成Raster,代码参考如下

outItRaster =r'%s\%s\_ItC%s.tif'%(outItPath,inputRaster[0:5],RightTempDatetime)  
outDatasetIt = driver.Create(outItRaster, ds.RasterXSize, ds.RasterYSize, 1, gdal.GDT\_Int16)  
outBandIt = outDatasetIt.GetRasterBand(1)  
outBandIt.WriteArray(tempIt, 0, 0)  
outDatasetIt.SetGeoTransform(geoTransform)  
outDatasetIt.SetProjection(proj)

修改代码过后，我再去检测内存消耗，每个循环由于tempIp、tempId、tempIdu、tempIr、tempIt增加的内存都被销毁了，在显式del之后，具体的内存检测代码我就不贴上来，就这样内存溢出的问题被解决了。