Subject: Re: question about 'Unable to allocate memory: to make array error' message (and I think there is enough memory)
Posted by chris_torrence@NOSPAM on Fri, 03 Oct 2014 04:09:20 GMT
View Forum Message <> Reply to Message

On Thursday, October 2, 2014 9:51:15 PM UTC-6, JP wrote:
> I solved my own problem.
>
>
>
> The issue was in the way I was defining my array which I wanted full of NaNs.
>
>
>
> Instead of
>
>   y= fltarr(sizeX[1], sizeX[2], sizeX[2]) & y[*]=NaN
>
>
>
> I did
>
>   y= fltarr(sizeX[1], sizeX[2], sizeX[2]) * NaN
>
>
>
> and saved memory (and time)
>
>
>
> same, more elegant:
>
>   y= MAKE_ARRAY(sizeX[1], sizeX[2], sizeX[2], /FLOAT, VALUE=NaN)
>
>
>
> I learned a lesson today.
>
>
>
> JP
>
>
>
>
>
>
>

> On Friday, 3 October 2014 11:54:24 UTC+10, JP wrote:
>
>> Guys, I'm trying to run a process in which I fall into a 'unable to allocate memory' issue, but I think I have enough memory and for some reason IDL doesnt' know it.
>
>>
>
>>
>
>>
>
>> I run my process and request 60GB of RAM.
>
>>
>
>>
>
>>
>
>> I create first a fltarr(16000000,19,6)
>
>>
>
>> if my numbers are right that's ~6.8 GB of memory
>
>>
>
>>
>
>>
>
>> I then try to create another fltarr of (16000000,19,19)
>
>>
>
>> again, if my numbers are correct, that is ~21.5 GB, total of ~28GB. When I try to do that IDL sends a "% Unable to allocate memory: to make array." error.
>
>>
>
>> Below my code and the output from the process.
>
>>
>
>> Any ideas?
>
>>
>

```
>>  Thanks
>
>>
>
>>
>
>>
>
>>  JP
>
>>
>
>>
>
>>
>
>>  pro test_memory_raijin
>
>>
>
>>    compile_opt idl2
>
>>
>
>>    nan= !Values.F_NAN
>
>>
>
>>
>
>>
>
>>    ; first make an array like the one is causing trouble
>
>>
>
>>    print, 'at the very beggining:' & help, /memory
>
>>
>
>>
>
>>
>
>>    x= fltarr(4000l*4000, 19, 6)
>
>>
>
```

```
>>    print, 'after creating x:' & help, /memory
>
>>
>
>>
>
>>
>
>>    ; now create another large array
>
>>
>
>>    sizeX = size(X)
>
>>
>
>>    y= fltarr(sizeX[1], sizeX[2], sizeX[2]) & y[*]=NaN
>
>>
>
>>    print, 'after creating y:' & help, /memory
>
>>
>
>>
>
>>
>
>>    ; get rid of y
>
>>
>
>>    undefine, y & help, /memory
>
>>
>
>>
>
>>
>
>>    ; now call medoid_2d
>
>>
>
>>    y= medoid_2d(x)
>
>>
>
```

```
>>    print, 'at the end:' & help, /memory
>
>>
>
>>
>
>>
>
>>  end
>
>>
>
>>
>
>>
>
>>  OUTPUT:
>
>>
>
>>
>
>>
>
>>
>
>>
>
>>  IDL Version 8.2.1 (linux x86_64 m64). (c) 2012, Exelis Visual Information Solutions, Inc.
>
>>
>
>>  Installation number: 237570.
>
>>
>
>>  Licensed for use by: ANU Supercomputer Facility
>
>>
>
>>
>
>>
>
>>  % Compiled module: TEST_MEMORY_RAIJIN.
>
>>
>
```

```
>>  at the very beggining:
>
>>
>
>>  heap memory used:    1181784, max:    9585806, gets:      734, frees:      292
>
>>
>
>>  after creating x:
>
>>
>
>>  heap memory used: 7297181928, max: 7297181928, gets:      735, frees:      292
>
>>
>
>>  % Unable to allocate memory: to make array.
>
>>
>
>>    Cannot allocate memory
>
>>
>
>>  % Execution halted at: TEST_MEMORY_RAIJIN   13
>
>>
>
>>     /home/599/jpg599/IDL_Scripts/raijin/test_memory_raijin.pro
>
>>
>
>>  %                      $MAIN$
>
>>
>
>>   ============================================================
==========================
>
>>
>
>>     Resource Usage on 2014-10-03 11:40:53.507223:
>
>>
>
>>   JobId: 7068191.r-man2
>
>>
```

```
>
>>   Project: k88
>
>>
>
>>   Exit Status: 0 (Linux Signal 0)
>
>>
>
>>   Service Units: 0.08
>
>>
>
>>   NCPUs Requested: 16    NCPUs Used: 16
>
>>
>
>>          CPU Time Used: 00:00:18
>
>>
>
>>   Memory Requested: 60gb     Memory Used: 16gb
>
>>
>
>>          Vmem Used: 30gb
>
>>
>
>>   Walltime requested: 00:10:00    Walltime Used: 00:00:19
>
>>
>
>>   jobfs request: 100mb    jobfs used: 1mb
>
>>
>
>>   ======================================================================
========================
```

Good catch JP. Using array indices like "*" can eat up a lot of memory because IDL needs to create an entire index array of 64-bit integers, which in your case ends up bigger than your original data!

One other point - sometimes (especially on Windows) your memory can become fragmented. So even though you might think you have lots of memory, if you allocate lots of tiny little arrays, the OS can allocate those all over the place, which makes it impossible to allocate a single huge chunk later on. So sometimes it is better to allocate your giant arrays first, before any smaller arrays.

-Chris