Subject: memory allocation for structure arrays
Posted by Ian Sprod on Wed, 29 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Hi,

I am trying to read a pretty large data file (40Mb) into IDL. The file
is 925,801 records, each 44 bytes long. I can describe the 44 bytes as a
data structure and then replicate this to make a structure array (albeit
a very large one). Then, in theory, reading in the data is a breeze.

The problem is that IDL runs out of memory trying to read in the file.
It seems that each line of the structure array is somehow requiring MORE
than 44 bytes of memory. Poking around with top and free shows that it
seems to be using ~312 bytes for each line instead. At this rate I can
only read in the first ~225,000 lines of the file.

Does anyone know exactly how IDL allocates memory for structures?

Should I be using an associative array to do this?

I am running IDL 5.0.2 on a Linux box with 128Mb of physical RAM and
twice that of swap space.

Thanks for any help or advice,

Ian

--

Ian E. Sprod
CIRES ian@ngdc.noaa.gov
NOAA/NGDC E/GC1 http://swat.ngdc.noaa.gov/~ian
325 Broadway 303-497-6284 (voice)
Boulder, CO 80303 303-497-6513 (fax)

Subject: Re: memory allocation for structure arrays
Posted by korpela on Thu, 30 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article <Pine.SUN.3.91.980430131547.324A-100000@demsyd.syd.dem.csiro.au>,
Peter Mason <peterm@demsyd.syd.dem.CSIRO.AU> wrote:

> Do you get the error during the READU call? (i.e., Not during the REPLICATE
> call?) If so then there's something very odd going on here.
> Can you read the first few (say 10) records correctly? If so, then I'd say
> that there might be a bug in IDL on Linux.

I seem to recall that IDL uses a lazy-malloc() which allocates memory without making sure there is enough available swap space. The swap space is allocated as a portion of memory is paged out for the first time. Therefore the out of memory error could during the READU call.

Eric
--
Eric Korpela | An object at rest can never be
korpela@ssl.berkeley.edu | stopped.
Click for home page.

Subject: Re: memory allocation for structure arrays
Posted by Ian Sprod on Fri, 01 May 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Eric and Peter,

Thanks for your detailed replies. I have plenty to work on now!

I should have been a little more specific - IDL fails with an out-of-memory error during the REPLICATE call. Not the READU call. So the lazy-malloc makes sense to me.

Just to recap I have a structure describing a 44 byte record. Some of the fields are strings, but I initialize the structure with strings the correct length. With smaller files I can read in the data without a problem. So I don't think strings are the problem (but it could be). Perhaps IDL is allocating a large amount of memory to each string field, just in case in needs to grow? This is not allowed with structures according to the manual, as once a structure's fields are defined they cannot be changed.

On a related point, when you do a :

help,data_structure,/structure

There is a field called "length". Is this the amount of memory in bytes allocated to the structure? I can't find this field defined in the hardcopy or online documentation.

Lets assume it is the structure memory allocation in bytes. Then we get odd results!

test = {byte1:0b,byte2:0b,byte3:0b}
help,test,/structure
** Structure <819499c>, 3 tags, length=3, refs=1:

```
  BYTE1       BYTE      0
  BYTE2       BYTE      0
  BYTE3       BYTE      0
```

>> Note the length is 3 - this looks OK for 3 bytes. What about for long-integers?

```
test = {long1:0l,long2:0l,long3:0l}
help,test,/structure
** Structure <8194ab4>, 3 tags, length=12, refs=1:
```

>> OK - this looks like what we would expect too 4 bytes per long integer. BUT try this one :

```
test = {long1:0l,byte2:0b,byte3:0b}
help,test,/structure
** Structure <8194e5c>, 3 tags, length=8, refs=1:
```

>> The length is 8 - I would expect 6 (4 + 1 + 1).

>> Now for strings it gets even worse :

```
test = {str1:'a'}
help,test,/structure
** Structure <8194ba4>, 1 tags, length=8, refs=1:

test = {str1:'aa',str2:'b'}
help,test,/structure
** Structure <8194e0c>, 2 tags, length=16, refs=1:
```

>> IDL seems to allocate 8 bytes per string field - no matter how long the string is initialized. I guess this is the "padding byte" problem Peter talks about in his email.

So for my supposedly 44 byte-long record I get a "length" of 104. If this is truly the length in bytes then REPLICATE would be looking for 104/44 or ~2.5 times more memory than I was expecting. This is pretty huge memory allocation when the file in question in 40Mb to begin with!

But then again maybe length is NOT the memory allocation in bytes :-)

I am still investigating this problem - please let me know if you can shed any more light on this. I'm going to take Peters suggestion and recode with bytarr instead of strings and hope that makes IDL behave a little better. Again thanks for your help already.

Ian

> In article <Pine.SUN.3.91.980430131547.324A-100000@demsyd.syd.dem.csiro.au>,
> Peter Mason <peterm@demsyd.syd.dem.CSIRO.AU> wrote:
>

>> Do you get the error during the READU call? (i.e., Not during the REPLICATE
>> call?) If so then there's something very odd going on here.
>> Can you read the first few (say 10) records correctly? If so, then I'd say
>> that there might be a bug in IDL on Linux.
>
> I seem to recall that IDL uses a lazy-malloc() which allocates memory
> without making sure there is enough available swap space. The swap
> space is allocated as a portion of memory is paged out for the first
> time. Therefore the out of memory error could during the READU call.
>
> Eric
> --
> Eric Korpela | An object at rest can never be
> korpela@ssl.berkeley.edu | stopped.
> Click for home page.

--

Ian E. Sprod
CIRES ian@ngdc.noaa.gov
NOAA/NGDC E/GC1 http://swat.ngdc.noaa.gov/~ian
325 Broadway 303-497-6284 (voice)
Boulder, CO 80303 303-497-6513 (fax)

Subject: Re: memory allocation for structure arrays
Posted by mallors on Sat, 02 May 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article <3549F646.51FF035E@ngdc.noaa.gov>,
 Ian Sprod <ian@ngdc.noaa.gov> writes:
>
> On a related point, when you do a :
>
> help,data_structure,/structure
>
> There is a field called "length". Is this the amount of memory in bytes
> allocated to the structure? I can't find this field defined in the
> hardcopy or online documentation.
> .
> .
> .
> test = {str1:'a'}
> help,test,/structure
> ** Structure <8194ba4>, 1 tags, length=8, refs=1:
>
> test = {str1:'aa',str2:'b'}
> help,test,/structure

> ** Structure <8194e0c>, 2 tags, length=16, refs=1:
>
> IDL seems to allocate 8 bytes per string field - no matter how
> long the string is initialized. I guess this is the "padding byte"
> problem Peter talks about in his email.

I was looking at the IDL Advanced Development Guide, and under the section "Mapping of Basic Types", an IDL_STRING is defined as follows:

```
typedef struct {
   unsigned short slen; /* Length of string */
   short stype;         /* Type of string */
   char *s;             /* Pointer to string */
} IDL_STRING;
```

I would guess this is how IDL defines its strings internally, so your structure example above would make sense, that is, each string would take 8 bytes: two for the length "slen", two for type "stype", and then the 4-byte pointer to the string (on most machines, but these are all machine-dependent).

For the other case you mention

> test = {long1:0l,byte2:0b,byte3:0b}
> help,test,/structure
> ** Structure <8194e5c>, 3 tags, length=8, refs=1:
>
> The length is 8 - I would expect 6 (4 + 1 + 1).

I think this has to do with how the structure is aligned in memory, as mentioned by Peter. Some hint of this can be seen as follows:

1. test = {long: 0L, b1: 0B, b2: 0B} => length = 8
2. test = {long: 0B, b1: 0B, b2: 0L} => length = 8
3. test = {long: 0B, b1: 0L, b2: 0B} => length = 12

On a 32-bit machine (4-byte words), case 1 uses one word to hold the long, and then another word to hold both of the byte variables. Similarly, in case 2, first word for the two bytes, another word for the long. But for case 3, the first word holds the first byte. Then we need another word to hold the long, because there is not enough room left in the first word. Then a third word is needed for the last byte, for a total of 3 words = 12 bytes.

-bob mallozzi

--
Robert S. Mallozzi
http://cspar.uah.edu/~mallozzir/

Subject: Re: memory allocation
Posted by davidf on Mon, 26 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Essa Yacoub (yacoub@cmrr.umn.edu) writes:

> I am assigning an array using the complex function
>
> rd=
> complex(mag_regress*cos(phase_regress),mag_regress*sin(phase _regress))
>
>>> 99% of the time this turns out to be a programming error >>>
>
> the code works, because I have successfully used it on several data
> sets of
> different sizes. In this particular case, the data size is much
> larger. I am running
> this on an sgi onyx 2 system with 2 gig of ram, so that should not
> be
> a problem. In fact I can open another session in another window and
> create bigger arrays with no problem. It actually crashed before
> this above statement
> and I zeroed an array I was no longer using and it continued a while
> longer.
> (using .con) before crashing again at a different line.

I'm going to guess that there are more of these large arrays
hanging around that you are also not cleaning up. When you
are finished with an array, you want to clear the memory that
is associated with it. This can be done by setting the array
to the value 0. Or it can be done more elegantly with my
UNDEFINE routine that you can find on my web page.

You also want to use the TEMPORARY function if you find
the same large array on both sides of the equal sign. That
is, instead of this:

array = array * 10

use this:

array = TEMPORARY(array) * 10

You might also try modularizing your code more. Many times
this problem comes about because people write huge main-level
programs instead of smaller procedures and functions.

> It is definitely a size problem but I don't understand
> why, or how I can clear up memory, or access more system memory....

To get more system memory allocated to your IDL process, it
helps to sacrifice your first-born child to the system administration
gods. But even if you do it when they are quite young and
innocent, it seldom helps. :-(

Cheers,

David

P.S. If your first-born is a teenager, forget it. The gods turn
those older sacrifices into administrators in charge of
program budgets. :-(

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: memory allocation
Posted by Essa Yacoub on Mon, 26 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David,

thanks for the reply...I am assigning an array using the complex
function

rd=
complex(mag_regress*cos(phase_regress),mag_regress*sin(phase _regress))

>> 99% of the time this turns out to be a programming error >>>

the code works, because I have successfully used it on several data sets of
different sizes. In this particular case, the data size is much larger. I am running
this on an sgi onyx 2 system with 2 gig of ram, so that should not be
a problem. In fact I can open another session in another window and create bigger arrays with no problem. It actually crashed before this above statement
and I zeroed an array I was no longer using and it continued a while longer.
(using .con) before crashing again at a different line. It is definitely a size
problem but I don't understand why, or how I can clear up memory, or

access more system memory....

thanks....
--
Essa Yacoub
University of Minnesota
Email: yacoub@cmrr.umn.edu

Subject: Re: memory allocation
Posted by davidf on Mon, 26 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Essa Yacoub (yacoub@cmrr.umn.edu) writes:

> I continue to get this error message,
> (pv-wave)
>
> Unable to allocate memory: to make array.
> Not enough space
>
> This is maybe a fragmented memory problem ?

99% of the time this turns out to be a programming error. But without more details, I could probably think of a dozen reasons you would get this message.

What are you trying to do when this message occurs? What kind of machine are you using? Are you trying to output an image in PostScript? Etc., etc.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: memory allocation
Posted by Essa Yacoub on Tue, 27 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Thanks all... The limit/unlimit command seems to be active
but I cannot change the memoryuse size, and datasize is unlimited.
the error seems to occur when I reach the memoryuse limit. The
sysadmin could also not change the size under root. He thought
he might need to reinstall some kernels. Anyhow, I will try
to adjust the code to work around this limit. The suggestion of
TEMPORARY would be very useful, however, does anyone
know if there is an equivalent pv-wave function, as it seems
this is only for IDL ?

thanks again.....

--
Essa Yacoub
University of Minnesota
Minneapolis, Minnesota 55455
Email: yacoub@cmrr.umn.edu

Subject: Re: memory allocation
Posted by Liam Gumley on Tue, 27 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
> You can try either limit (csh or tcsh) or ulimit (sh). By itself,
> "limit" produces this output for me on an alpha machine:
>
> [33]> limit
> ...
> datasize 131072 kbytes
> ...
>
> If you get a small number like I did, then you can try increasing it,
> like so,

>
> limit datasize 2048M # 2048 megabytes = 2 gigabytes
>
> Usage may vary, depending on your shell and Unix version. If you
> can't increase it yourself, it's time to abase yourself to your
> sysadmin.

The 'unlimit' command may also prove useful. Try

% limit
% unlimit
% limit

and the limits should have increased. I always put 'unlimit' in my
.cshrc or .profile login script. In addition, you might want to check
the virtual memory allocation on your system using 'swap'. On my SGI
system, I get

% /sbin/swap -ln
 # path pri pswap free maxswap vswap
 1 /dev/swap 0 512.00m 512.00m 512.00m 0.00k

which shows I have 512MB of swap space. Check with your sysadmin if this
number seems small.

Cheers,
Liam.

--
Liam E. Gumley
Space Science and Engineering Center, UW-Madison
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: memory allocation
Posted by Craig Markwardt on Tue, 27 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:
>
> Essa Yacoub (yacoub@cmrr.umn.edu) writes:
>
>> I am assigning an array using the complex function
>>
>> rd=
>> complex(mag_regress*cos(phase_regress),mag_regress*sin(phase _regress))
>>
>>>> 99% of the time this turns out to be a programming error >>>

>>
>>   the code works, because I have successfully used it on several data
>> sets of
>>   different sizes. In this particular case, the data size is much
>> larger. I am running
>>   this on an sgi onyx 2 system with 2 gig of ram, so that should not
>> be
>>   a problem. In fact I can open another session in another window and
>> create   bigger arrays with no problem. It actually crashed before
>> this above statement
>>   and I zeroed an array I was no longer using and it continued a while
>> longer.
>>   (using .con) before crashing again at a different line.
>
> I'm going to guess that there are more of these large arrays
> hanging around that you are also not cleaning up. When you
> ...

David makes some good suggestions for being memory efficient in your
IDL programs. However, if you are running on a Unix system you may be
running into the memory limit for a single process. Memory footprints
for processes are typically limited to prevent a single task from
hogging all of the resources, but often this is what you want!

You can try either limit (csh or tcsh) or ulimit (sh). By itself,
"limit" produces this output for me on an alpha machine:

[33]> limit
...
datasize   131072 kbytes
...

If you get a small number like I did, then you can try increasing it,
like so,

limit datasize 2048M   # 2048 megabytes = 2 gigabytes

Usage may vary, depending on your shell and Unix version. If you
can't increase it yourself, it's time to abase yourself to your
sysadmin.

Good luck,
Craig

--
 -- -------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu