Subject: memory leak with HDF5?
Posted by peter.albert@gmx.de on Thu, 21 Jul 2005 14:30:29 GMT
View Forum Message <> Reply to Message

Hi everybody,

I am new to this group, and I am experiencing a strange memory leak when reading HDF5 files with IDL 6.1 (on a IBM-AIX machine). If I am running the following code fragment, with "files" being an array with filenames of HDF5 files, which all contain a "Data/Data1" dataset:

```
for i = 0, n _files - 1 do begin
   file_id = h5f_open(files[i])
   nd = h5g_get_nmembers(file_id, "Data")
   dataset_id = h5d_open(file_id, "Data/Data1")
   dataset = h5d_read(dataset_id)
   h5d_close, dataset_id
   h5f_close, file_id
endfor
```

then the core image of the IDL process increases by appro. 400k in each loop, which means that after a sufficent large number of files I get the follwoing error

```
% Unable to allocate memory: to make array.
  Not enough space
```

I have to admit that I do not exactly know what "core image of the IDL process" actually means, but that's what the manpage of the Unix "ps" command tells me ... :-) I did put the following line before the "endfor" statement:

```
spawn,  "ps axu | grep palbert | grep idl | grep -v grep"
```

which actually showed me, among other info, well, the size of the core image. And it just constantly increased.

I also put a "help, /memory" there, of course, but this number kept constant, so it is not IDL saving more and more variables or so.

Now, the funny thing is, if I exclude the

```
nd = h5g_get_nmembers(file_id, "Data")
```

command, then the core size increases much more slowly.
I have no idea what is going on here.

Moreover, if I open the same file again and again, nothing happens.

??? I am completely lost.

I would really like to run my code without crashing after a few hundred files, so if anyone has an idea what is happening here, any comment would be greatly appreciated.

Best regards,

Peter

Subject: Re: memory
Posted by Karl Schultz on Wed, 30 Aug 2006 16:46:54 GMT
View Forum Message <> Reply to Message

On Wed, 30 Aug 2006 10:06:01 -0600, Jean H. wrote:

> Hello all,
>
> Is there a way for IDL to find out the amount of available memory?
>
> Also I am not too sure to understand how IDL is managing the virtual
> memory. In the help file, they say that IDL would automatically write
> information on the disk if the physical memory gets full. However, I
> have a function that crashes (i.e. % Unable to allocate memory.) as soon
> as my physical memory is totally used. In this function, I have to carry
> a few quite large arrays that are used only once in each loop
> iteration.. so technically speaking, they could be written on the disk
> instead of being kept in memory!
> I have 2G of physical memory, and the same amount of virtual memory (for
> a suposedly nice total of 4G then).
>
> IDL> print, !version
> { x86 Win32 Windows Microsoft Windows 6.2 Jun 20 2005 32 64}
>
> Any thoughts would be greatly appreciated!
> Jean
>
> PS: Having to add some loops in your code because the loopless one takes
> too much memory is more painfull than pulling your hairs when writing
> the "optimized" code..

This topic has been covered a great deal in this newsgroup over the years.
You might want to check the archives if this summary isn't enough.

1) 32-bit Windows reserves half of the 4G address space for the OS. I'm not sure, but that may have improved to a 3G/1G app/os split in XP.

2) IDL does not manage its virtual storage. It lets the OS do it. As physical memory gets committed, the OS will page out blocks of memory that have not been used recently to disk.

3) Most memory allocation failures on Windows are due to virtual address space fragmentation. (This is completely independent of paging) The problem is that sometimes there is not enough free CONTIGUOUS virtual address space to satisfy a request for a large allocation, even though there are more than enough smaller free blocks around to fill the request. You may find that you can't allocate a 500MB array, but you can allocate 4 or 5 200MB arrays.

These are all just characteristics of 32-bit Windows. You'll either have to redesign your algorithms to not rely on such large allocations or move to a 64-bit OS. A 64-bit address space suffers less from fragmentation.

Some folks have reported better success by moving to 32-bit Linux, because the virtual address fragmentation problem is not as severe. Windows divides up the user portion on the virtual address space into areas for specific uses and can load things in the middle of large free blocks. Both of these actions fragment the address space.

Karl

Subject: Re: memory
Posted by Jean H. on Wed, 30 Aug 2006 20:13:13 GMT
View Forum Message <> Reply to Message

> These are all just characteristics of 32-bit Windows. You'll either have
> to redesign your algorithms to not rely on such large allocations or
> move to a 64-bit OS. A 64-bit address space suffers less from
> fragmentation.
>
> Some folks have reported better success by moving to 32-bit Linux, because
> the virtual address fragmentation problem is not as severe. Windows
> divides up the user portion on the virtual address space into areas for
> specific uses and can load things in the middle of large free blocks.
> Both of these actions fragment the address space.
>
> Karl

Thanks for this point... I guess it is the source of my problems since the first thing my application does it to "reserve" most of the memory

by creating a huge array and setting it back to 0..
Also each array requires about 400 Mbytes ... I don't have a clue why
none of them is sent to the virtual memory, even with a 4G of virtual mem.

Jean

Subject: Re: memory
Posted by Karl Schultz on Thu, 31 Aug 2006 16:05:33 GMT
View Forum Message <> Reply to Message

On Wed, 30 Aug 2006 14:13:13 -0600, Jean H. wrote:
> Thanks for this point... I guess it is the source of my problems since
> the first thing my application does it to "reserve" most of the memory
> by creating a huge array and setting it back to 0..
> Also each array requires about 400 Mbytes ... I don't have a clue why
> none of them is sent to the virtual memory, even with a 4G of virtual mem.

http://en.wikipedia.org/wiki/Virtual_memory

may be useful.

Karl

Subject: Re: Memory Leak
Posted by David Fanning on Mon, 27 Jul 2009 22:21:24 GMT
View Forum Message <> Reply to Message

wlandsman writes:

> IDL> print,!version
> { x86 linux unix linux 7.0 Oct 25 2007 32 64}
>
> IDL> print,(memory())(3)
> 778166
> IDL> test1 & print,(memory())(3)
> 3673008
> IDL> test & print,(memory())(3)
> 4273023
> IDL> test & print,(memory())(3)
> 4873023

I see a little different pattern on Windows:

IDL> test & print,(memory())(3)
4070207

```
IDL> test &  print,(memory())(3)
     5070207
IDL> test &  print,(memory())(3)
     6070207
IDL> test &  print,(memory())(3)
     7070207
```

Not sure what to make of this.

Cheers,

David

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Memory Leak
Posted by chris_torrence@NOSPAM on Tue, 28 Jul 2009 18:44:14 GMT
View Forum Message <> Reply to Message

Good catch guys. This is indeed a bug. You can actually reproduce it
using just a simple string array:

```
pro cr56738_write
  a = bytarr(100000)
  openw,lun,'c:/test.dat',/get_lun
  writeu,lun,a
  free_lun,lun
end

pro cr56738_read
  openr,lun,'c:/test.dat',/get_lun
  out = replicate(' ', 100000)
  readu,lun,out
  free_lun,lun
end

cr56738_read & print,(memory())[3]
     3005089
cr56738_read & print,(memory())[3]
     4005080
cr56738_read & print,(memory())[3]
     5005080
```

This has been logged as CR56738, and will be fixed in the next IDL

release.

As a workaround, you could read the data as bytes, and then convert the result to a string afterwards.

Cheers,
Chris
ITTVIS
