Subject: Memory Headaches
Posted by crono15m on Thu, 01 Aug 2002 15:28:18 GMT
View Forum Message <> Reply to Message

I have a problem which seems to be getting more and more desperate and I'm hoping someone here may have some insight into it.

I have a situation where there is no (practical) way around loading very large files into memory as IDL variables. The problem is that sometimes they are too big and I run out of memory. I've tried increasing the amount of virtual memory in the system but IDL doesn't seem to know how to take advantage of all the virtual memory available so it's probably capped.

Eric Korpela's VARRAY solution (http://albert.ssl.berkeley.edu/~korpela/mmap/) seems like a beautifully elegant solution to my problem since I could just map the file and IDL wouldn't know the difference. However, I'm running win2k and Eric's program is only for Unix. I've looked at the source for his program to see if maybe I could modify it to run in windows but unfortunately my Unix/WinAPI/C skills seem to be lacking for the job.

If it were just me running IDL I would consider running a Unix emulator but if all goes well my analysis package could be running at several colleges and I'd rather keep the solution as unobtrusive to the user as possible. This may be asking too much. I don't know.

So basically my questions are these,
Does anyone know of a way to allow IDL to use more (maybe 1GB) virtual memory in windows?
Does anyone know of a C library that can allow windows to emulate Unix commands such as mmap and ftruncate?
Any other suggestions?

Thanks for any help,

Ben Hilldore
Hope College Nuclear Research Group

Subject: Re: Memory Headaches
Posted by Karl Schultz on Sat, 03 Aug 2002 23:17:09 GMT
View Forum Message <> Reply to Message

I've wondered about this too and decided to dig into it a little today.
Read on.

"Mark Rivers" <rivers@cars.uchicago.edu> wrote in message

news:MGF29.101$15.30101@news.uchicago.edu...
> Ben <crono15m@aol.com> wrote in message
> news:fbb3dcd9.0208020603.34689f8f@posting.google.com...
>> Thank you everyone who replied.
>>
>> I would love to add more RAM to my system. Unfortunately I am limited
>> at 512 by my motherboard.
>>
>> I then ran the same test on a machine running NT4 Sp6 (512MB ram)
>>
>> The largest array on that system was around 528,000,000l independent
>> of how high I set the swap. (assuming the swap was big enough)
>>
>> This makes me think that the limit probably is OS dependant.
>
> This is consistent with my experience with Windows NT. I have 2 machines
> with 1 GB of RAM and 3GB of swap space each, and the most memory IDL can
> allocate is 1GB. This is also what RSI also told you was the NT limit.

I checked and we (IDL) certainly do not cap the memory request. I followed
the request all the way down to the base heap allocator in NT and it refuses
a request over 1G.

Keep in mind too, the difference between the maximum request size and the
total allocation limit. It may just be that the biggest contiguous free
space is less than 1G in size. You are probably able to make many smaller
requests that add up to more than 1G. But I know that's not the main issue
here. Read on. I promise, it gets better.

> Note however, that this is inconsistent with every piece of Windows NT
> documentation I have found, which states that a single NT process can
> allocate 2GB (not 1GB).

This is because NT uses the top 2G for the kernel and the bottom 2G for
apps.
I suppose that checking the top bit of the address was a handy way to split
the space.

I think that NT Server got changed to allow 3G user.

> In my case the difference would be very
> significant, and if IDL could access 2GB I would go buy a new computer
with
> 2GB of RAM in a heartbeat. (My application is 3-D tomography data sets,
> where being able to quickly look at slices in any direction is critical,
and
> makes re-reading the desired subset data from disk totally impractical.
>

> Of course, I have not yet taken the 10 minutes it would take to test the
> following C program
>
> main()
> {
> char *t = malloc(unsigned long(1024*1024*1024*1.5));
> }
>
> to see if a C program can allocate more than 1GB of memory as the NT
> documentation indicates, or whether a C program is also limited to 1GB.
> Just laziness!

OK, I tried this experiment with a Windows app (not a console app) and found that I could allocate more than 1 G. So what makes IDL different???

IDL loads a lot of DLL's. Each DLL has a preferred loading address, which can cause NT to plop a DLL anywhere the DLL wants in VM. If there are conflicts with other DLL's, NT will relocate the DLL's so that they all get loaded.

I suspect that some DLL that IDL loads (or a DLL that another DLL loads) is requesting a load address which badly frags the virtual memory. I didn't dig deeply enough to find the culprit(s), and I suppose I will eventually do that. But you could imagine a DLL sitting in VM exactly at the 1G address, which would make it impossible to allocate anything bigger than 1G in one chunk.

So, what to do?

There is a tool that can be used to rebase DLL files. There may be more than one, but the one I used is called EDITBIN and is shipped with the Microsoft Visual C++ compiler/studio. Here's what I did:

1) Make sure you are not running IDL
2) go to your IDL bin directory and make a backup copy of the entire bin.x86 directory.
3) go into the bin.x86 directory and issue:

    EDITBIN /REBASE idlde.exe *.dll

It will grind for a few seconds and then finish. It worked OK for me on the 5.5 code. If it complains about a DLL, rename it to .DLX, try again, and then rename it back.

Then go ahead and run IDL and try your large allocation again. I managed to get a *little* over 1G, but that may be because there are still some DLL's outside of the IDL bin directory that are not loading very nicely, or I had too much other stuff in VM or who knows. Other people may get different

results.

Another nice side-effect of doing this is that NT won't have to relocate IDL DLL's anymore, which may decrease IDL load time. The downside is that you may get VM holes for the DLL's (DLM in IDL-speak) that do not get loaded until you need them. More study required here.

Please consider this idea as highly experimental and not as an official solution from RSI.

If you want to know more about the issue, google for "DLL relocation" to find articles at the detail level you want.

I'll see what can be done about easing this problem in the next IDL release.

Karl
RSI
