Subject: Re: largest array, most memory accessible
Posted by Karl Schultz on Thu, 14 Oct 2004 00:22:06 GMT
View Forum Message <> Reply to Message

"R.G. Stockwell" <noemail@please.com> wrote in message
news:2t5g66F1ghte9U1@uni-berlin.de...
> I was following some of the other threads about memory size,
> and wanted to see what the options are for a new computer I am
considering.
> (I am looking at getting a dell dimension or something in that price range
and wondering
> how much ram is useful. The main thing to spend money on is ram, even if
I get a slower
> processor)
>
> My dream would be to have an array of complex variables of size [1024,
1024, 1024,512],
> (four terrabytes of ram?) but I guess I can scrape by with arrays of
[128,128,128,64] perhaps.
> I cannot test out these values on my current computers.
>
>
> 1) What is the largest array in IDL 6.1 on a winXP machine?
> Is this 2 Gb?
>
> 2) What is the largest amount of ram IDL can access, i.e. can I
> make multiple copies of the array in question 1?
> Is this limit also 2Gb or can IDL grab the full 4gbs?

One place I can point you to is:

http://tinyurl.com/5uhom

aka:

http://groups.google.com/groups?hl=en&lr=&newwindow=
1&safe=off&threadm=e5SG4w7dDHA.2312%40TK2MSFTNGP12.p
hx.gbl&rnum=3&prev=/groups%3Fhl%3Den%26lr%3D%26newwi
ndow%3D1%26safe%3Doff%26q%3DwinXP%2Bcontiguous%2Baddress%2Bs pace

Other notes:

1) The amount of RAM in your machine doesn't really limit the size of the
problem you are trying to solve. More RAM means less paging and you will
get your problem solved faster. If you have a small amount of RAM and a lot
of virtual storage (large page file, a big enough disk, and an OS that
supports a large enough virtual address space) you can still finish, but it

will take a long time. If the entire problem fits into RAM, then it will cruise, and that's why people get a lot of RAM.

2) Win XP can be configured to have up to 3GB of virtual storage. Usually, it is 2GB. The 32-bit processors can address up to 4GB, but the Windows (NT) architecture usually reserves the upper 2GB for the kernel. There is plenty of documentation on the net about setting the user space to 3GB, hence setting the kernel space to 1GB. So, IDL can never grab the full 4GB. At best, it can get only part of the 2GB or 3GB that is the user virtual address space.

3) And here is the most important part, which is covered by the URL I noted above. The limiting factor on the largest array you can allocate in IDL is the largest piece of *contiguous virtual* address space that is available. So, for example, if Windows, IDL, and a lot of its required libraries occupy the first 0.25GB of virtual address space, and Windows also grabs some of the upper user virtual address space, say another 0.25 GB, then, at best, you will be able to allocate a 1.5GB array (assuming the std 2GB available). If a small DLL or some other small thing gets loaded in the middle of the virtual address space, say at the 1GB mark, then the virtual address space is fragmented and now you can only get two 750 MB arrays. This can happen, for example, if your IDL program allocates a few large arrays and a small array that happened to fall in between two large arrays. If you free the two large arrays, thinking that you will be able to allocate a single array that is the combined size of the two arrays you just freed, then the attempt will fail unless there is a big enough chunk of virtual address space someplace else.

Anyway, I seem to remember some discussion here awhile ago saying that people could get 1.2 or 1.3 GB arrays on Windows machines. I doubt that the numbers today are much different, unless you throw the 3GB switch. Perhaps trying the 3GB configuration is worthwhile for you, or use a 64-bit OS.

Hope this helps,
Karl
