Subject: Re: Minor IDL code changes cause large slowdowns elsewhere in code
Posted by R.G.Stockwell on Wed, 10 Oct 2007 19:05:38 GMT
View Forum Message <> Reply to Message

"cedric" <cedric@barrodale.com> wrote in message
news:1192042534.817071.151580@19g2000hsx.googlegroups.com...
> I have observed a problem in an IDL timber supply model that arose
> after having made some changes to use tables instead of computations
> in order to free up some memory space and to circumvent some involved
> computations.

offhand I would say that it is a memory problem. If you are causing any swapping to disk, that is a killer. Could be due to fragmentation - try allocating
huge blocks of memory at the start, thay may help.

Also, are you sure you did not introduce a memory leak with the modifications?

Cheers,
bob

Subject: Re: Minor IDL code changes cause large slowdowns elsewhere in code
Posted by Haje Korth on Wed, 10 Oct 2007 21:04:52 GMT
View Forum Message <> Reply to Message

Any difference if you try the comparison with integer numbers assigned to the tree types? Haje

"cedric" <cedric@barrodale.com> wrote in message
news:1192042534.817071.151580@19g2000hsx.googlegroups.com...
> I have observed a problem in an IDL timber supply model that arose
> after having made some changes to use tables instead of computations
> in order to free up some memory space and to circumvent some involved
> computations. Following these changes, there was a general four-fold
> increase in execution times, even for those sections of code that were
> unaffected by the change. After doing some analysis, I found that
> this was at least partly due to large increases in times for
> operations involving manipulating string fields in a vector of
> structures (with, say, 50,000 elements).
>
> For example, we have a string vector of the form
> (*(*unit[i]).layer).species, where "unit" is a pointer to a vector of
> large (30 MByte) "unit" structures with multiple tags, one of which is
> a pointer to a vector of "layer" structures, and where each "layer"

> structure has "species" (a string) as one of its tags. Then commands
> of the form
>
> z = uniq ((*(*unit[i]).layer).species, sort
> ((*(*unit[i]).layer).species)) , or
> subs = where ((*(*unit[i]).layer).species eq 'PINE')
>
> take much longer than with the original version (with the same
> elements in the vector).
>
> I have some work-arounds to recover some of the speed, but the
> question is what is really going on here, where minor changes in the
> code can cause large changes in the timing behavior of procedures that
> are outside the code that was changed? Is there some memory
> fragmentation issue? If so, how can this be overcome? (BTW, the
> memory footprint of the code with tables is actually 40% smaller than
> the original!) If anyone has any experience with something similar, I
> would really appreciate their insights here.
>

Subject: Re: Minor IDL code changes cause large slowdowns elsewhere in code
Posted by cedric on Thu, 11 Oct 2007 19:06:28 GMT
View Forum Message <> Reply to Message

On Oct 10, 2:04 pm, "Haje Korth" <haje.ko...@nospam.jhuapl.edu> wrote:
> Any difference if you try the comparison with integer numbers assigned to
> the tree types? Haje
>

Thanks, Haje. With integers or floats the manipulations are much
faster. I guess that this is probably due to the need to do separate
memory allocations for each element of the string vectors during the
processing. I could change the codes to be integer, but am still
perplexed that the identical string vector manipulations were far
faster with the original code before the changes...

Subject: Re: Minor IDL code changes cause large slowdowns elsewhere in code
Posted by cedric on Thu, 11 Oct 2007 21:52:55 GMT
View Forum Message <> Reply to Message

On Oct 10, 12:05 pm, "R.G. Stockwell" <noem...@please.com> wrote:
> "cedric" <ced...@barrodale.com> wrote in message
>
> news:1192042534.817071.151580@19g2000hsx.googlegroups.com...
>

>> I have observed a problem in an IDL timber supply model that arose
>> after having made some changes to use tables instead of computations
>> in order to free up some memory space and to circumvent some involved
>> computations.
>
> offhand I would say that it is a memory problem. If you are causing any
> swapping to disk, that is a killer. Could be due to fragmentation - try
> allocating
> huge blocks of memory at the start, thay may help.
>
> Also, are you sure you did not introduce a memory leak with the
> modifications?
>
> Cheers,
> bob

Thanks for your response, Bob. I have tried pre-allocating large memory, and even rebooting, pre-allocating, and running with no other user processors. No improvement, unfortunately.

As for a memory leak, I could be wrong but I don't see how adding a data structure consisting of a few tables (30 x 50,000 - no pointers) could be a source of a memory leak. Also, looking at task manager, the memory seems to hold about constant after the first initial load (800 MB). Of course, I can't really be sure here. Any suggestions about how to detect a memory leak in an IDL process?
