Subject: Minor IDL code changes cause large slowdowns elsewhere in code
Posted by cedric on Wed, 10 Oct 2007 18:55:34 GMT
View Forum Message <> Reply to Message

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.