Subject: Re: compile a routine wich inlude a commun
Posted by Maarten[1] on Tue, 24 Jan 2006 10:04:01 GMT
View Forum Message <> Reply to Message

JD Smith wrote:

> I think he means pointers are a kludge for extensible arrays.

I think you can read minds, that is what I meant.

> *But*, and this is a big but, all that flexibility comes at
> some real cost in speed, which grows with the data size, perhaps
> non-linearly.

Understood, and at times: accepted.

[snip, reasons to use IDL, and if speed is added, the "flexible"
languages may turn into a two-lobed "mess" as well]

> 6) Want to share code which just runs with colleagues, avoiding the
> package dependency and moving target problems of roll your own
> solutions like Python + numarray, or numeric, or numpy, ...
>
> Of course, this should include a footnote of {Rich colleagues who can
> afford IDL licenses}.

I can see that, and it is one of the reasons I haven't switched to
Python yet. The footnote adds one of my private adversions: at work I
can use IDL, good ideas occuring at home may go wasted.

[snip: links to IDL <---> python. thanks]

> Another thing you'll notice with most of these packages (and, sadly,
> even GDL) -- plotting is typically a compromise, borrowing a
> pre-existing package like GnuPlot, or matplotlib, not very cleanly
> integrated. It's a real pain to integrate decent graphics in a
> compatible, cross-platform way. I think this problem will eventually
> be solved, but for now, if I send you a Python+numpy+matplotlib
> script, it probably wouldn't run out of the box.

Agreed. Cairo Graphics (http://cairographics.org/introduction) seem
promising, but "not there yet". For my thesis I didn't like the output
of any of the graphics packages, so the final output was created in
MetaPost. Today I might have choosen PyX.

[on the Drizzling page at David's site]

> As one of the perpetrators of that page, I have to agree, these

> examples (and many of the IDL Way) are not terribly obvious. Some
> have maintenance concerns, to be sure. But, they enable you, in a
> typeless language, to obtain the kind of speed of operation on large
> (many MB to many GB) piles of data that are simply otherwise unheard
> of.

I think we disagree where the balance between readability and execution
speed lies.

> Moreover, a elegant Python Drizzle would probably run 10x slower even
> than the straightforward loop-based IDL drizzle. At some point, you
> give up and write it quite simply in C, spending 95% of the time and C
> code figuring out how to communicate the results back with IDL. So, I
> agree with the original poster that the algorithms mentioned, among
> many others in IDL, are not at all transparent, while simple versions
> of the same are not at all fast. However, in my experience, this is
> the price you pay in the tradeoff of elegance and raw speed.

I prefer my "normal" everyday use to be elegant, and when I really need
the speed (after profiling, no need for premature optimization), an
easy route to C, Fortran, (assembly, no, I'm not that nuts) is
appreciated.

> I think if RSI wants to do one thing to move the tradeoff forward,
> they should take MAKE_DLL and vastly expand its scope, allowing you to
> trivially stick *simple* bits of C-code callout which operate directly
> on IDL data in memory.

I have some experience with inserting C code into Labview. There are
several methods to do so: one truly integrated, and I think it is about
as hard to do as in IDL, and probably harder to debug. Another method
is to call into a shared library. As long as you leave the memory
allocation of anything that gets communicated back into (Labview/IDL)
to the calling application, it is near trivial - just be careful to
clean up after yourself.

It wasn't at the level where pyrex lives, but easy enough, and a
compilable shell for debugging was easy enough to generate.

I hope IDL will get to that point - or maybe I've not found the right
sample code to do this (especially on a 2D array). A pity that SWIG
doesn't support IDL, it might make some things quite a bit easier.

> Python has several projects aiming to do just
> this, and if any of them become standard, this may change the
> scientific coding landscape significantly.

I think that is a given, it is only a matter of time. Another project

to keep an eye on is PyTables: transparent persistent storage of arrays
in an HDF file with a decent speed.

Maarten