Subject: IDL vs PV-WAVE
Posted by dan on Wed, 24 Aug 1994 14:56:51 GMT
View Forum Message <> Reply to Message

We are currently looking for upgrading our old PV-WAVE v4.00
and have had nice bids from both IDL and VNI (or whatever
they are called these days). I understand (from FAQ) that
IDL and PVWAVE are really the same product up to about
v4.00 but since then things changed.

Can anyone enlighten me and explain what are the major
differences in the latest IDL and PV-WAVE and (perhaps)
which is better (not trying to create a religious war,
really).

Thanks in advance.

dan

Subject: Re: IDL vs PV-WAVE
Posted by thompson on Fri, 26 Aug 1994 14:35:18 GMT
View Forum Message <> Reply to Message

tonyg@hdos.hac.com (Tony Grusczak) writes:

> ... Many people I know use C or Fortran to do their heavy data processing
> (which is performed much faster in a compiled program than when done in IDL or
> PV-WAVE) then pass it on to the graphics package for its display capabilities.

(rest deleted)

It's not really *quite* true that C and Fortran are much faster then IDL or
PV-Wave. It really depends on the problem. If it can be expressed in array
notation, then IDL and its cousin PV-Wave perform quite competitively with C
and Fortran. For example, I find no perceptible difference between IDL and
Fortran in doing non-linear least-squares fits, even on slow machines like a
MicroVAX II.

The kicker is usually loops. Procedures in IDL that require looping generally
perform slowly because it's an interpretive language. I've often found that
IDL novices who are more used to compiler languages use loops to do things that
can be done more elegantly in IDL without them (I did it myself when I started
out.) Admittedly, some of the tricks one can use in IDL to avoid loops are not
exactly obvious, although most are quite straightforward and often easier to
understand than the loops in compiler languages.

I haven't tried RPC, but I have used the reverse, which is to use CALL_EXTERNAL

to call C and Fortran routines from IDL. That way, I can do most of the programming in IDL, which is easier than C or Fortran, and can make use of nice features like rapid prototyping and the IDL widget interface, and still use C or Fortran for when one has to do real number crunching. It makes more sense to me to have IDL as the front end, and the compiler languages as the back end. Of course, if one has a heritage system that one needs to integrate with a package like IDL, then it does make sense to use RPCs to do it the other way around.

Maybe the confusion has to do with the platform being used. RPCs, and their inverses CALL_EXTERNAL and LINKIMAGE, have been around for some time on many Unix and VMS platforms, I believe that it wasn't supported under Windows until more recently. Also, some platforms, like DEC Ultrix workstations, still don't support things like CALL_EXTERNAL.

Bill Thompson

Subject: Re: IDL vs PV-WAVE
Posted by peter on Fri, 26 Aug 1994 16:40:12 GMT
View Forum Message <> Reply to Message

Mitchell R Grunes (grunes@imsy1.nrl.navy.mil) wrote:

: Also, IDL's TEMPORARY function could be a life saver if you have
: limited memory space, especially if you want to run fast. If PV-Wave
: has an equivalent, I can't find it. (ANYONE?)

I asked PV-Wave tech support about this (after moving companies and changing from IDL to PV-Wave). They said they don't have it and have no intention of implementing it. I used to use it all the time to implement a poor man's stack of data sets (using TEMPORARY allowed stack pushes and pops without actually copying any data), so I was rather annoyed when it was not available.

Peter

Subject: Re: IDL vs PV-WAVE
Posted by cs61a-ab on Sun, 28 Aug 1994 08:16:46 GMT
View Forum Message <> Reply to Message

In article <33kuj6$ktt@paperboy.gsfc.nasa.gov>,
William Thompson <thompson@orpheus.gsfc.nasa.gov> wrote:
<stuff deleted>
> The kicker is usually loops. Procedures in IDL that require looping generally
> perform slowly because it's an interpretive language. I've often found that

> IDL novices who are more used to compiler languages use loops to do things that
> can be done more elegantly in IDL without them (I did it myself when I started
> out.)
> Admittedly, some of the tricks one can use in IDL to avoid loops are not
> exactly obvious, although most are quite straightforward and often easier to
> understand than the loops in compiler languages.
<more stuff deleted>
>
> Bill Thompson

What are some of these "tricks" you are talking about? I am starting to learn
IDL and if there are some things I can avoid doing or make sure I do,from the
get go, that would be great to know.

Thanks,

Michel

PS: Other than the 600 pages of online help that IDL has, is/are there
any other sources (online/inprint) that would be helpful to learn
IDL?
