Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by stevenj on Wed, 16 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

steinhh@ulrik.uio.no (Stein Vidar Hagfors Haugan) wrote:
>> - If the FFTW (which is free) outperforms the native FFT in IDL,
>> why don't RSI use that implementation? Is this a silly question?
>
> Since FFTW is free under the Gnu Public License (GPL), RSI would
> have to *give away* IDL under the GPL license in order to include
> FFTW as part of IDL (I think). Not a good business practice! There
> are two options, however:

That's not entirely accurate, although I agree with your conclusion that RSI is unlikely to use FFTW under the GNU license.

It *is* legal to sell GPL'ed software (e.g. Red Hat Linux); you can also sell technical support, documentation, etcetera. However, you have to allow unrestricted redistribution and you also have to make your entire program's source code available under the GPL, so most companies are unwilling to use this business model. (In practice, you can't charge very much for the program itself if it is GPL'ed, although you can charge more for support.)

> 1. RSI could produce a wrapper for FFTW and make it available
> through their web site so the user could make a dynamically loadable
> module (and the user would have to fetch the FFTW separately). AFAIK
> this would mean that FFTW is not "sold as a part of IDL".

Nope, you can't get around the GPL in this way. To quote R. Stallman of GNU, "A GPL-covered plug-in that is designed to be combined with [only] a non-free master program is a form of combined work, and a violation of the GPL." (The original authors can make an exception allowing their code to be used in such a plugin, but no one else can do so.)

(Otherwise, the GPL would essentially devolve to the LGPL--you could link any GPL'ed code you wanted into a non-free program just by making it a "plugin." For more info, do a search on Dejanews for: ~a (rms@santafe.edu) & ~g (gnu.misc.discuss) & "Plug-ins")

You can make such a plugin for your own use, but you can't distribute it. (In any case, you are probably right in that such a plugin wouldn't be widely useful unless it came with IDL.)

> 2. The FFTW site mentions the possibility of non-free licenses. This
> may be an idea to look into (after all, they are paying licences
> for routines from Numerical Recipes).

Yup, in the case of FFTW, unrestricted licenses are available, for a fee,
from MIT. That's up to RSI, though.

> However, I'm not so sure the *algorithm* of IDL's FFT is so bad,
> I suspect (lacking) compiler optimizations to play a part here.
> This is probably also the reason for the drop in performance
> experienced in going from Fortran to C source code (Fortran
> code is easier to optimize - the compiler can make stronger
> assumptions on what is going on).

I tend to think that this is overstated, since the aliasing problems of C
can be avoided by an alert programmer (you just have to store dereferenced
pointers in local variables for performance-critical regions). However,
uncareful translations of Fortran programs do have a tendency to suffer.

Cordially,
Steven G. Johnson

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by roy.hansen on Wed, 16 Sep 1998 07:00:00 GMT

View Forum Message <> Reply to Message

In article <6to8cubgm1@readme.uio.no>, steinhh@ulrik.uio.no (Stein Vidar Hagfors Haugan)
wrote:
>
> In article <35ff8db4.0@d2o203.telia.com> roy.hansen@triad.no
> (Roy E. Hansen) writes:
>> We did a small comparison of the FFT performance in IDL 5.1.1
>> compared with the Matlab 5.2 version for a PII-400 with Win-NT,
>> and found that Matlab was approx 4 times faster. We also found
>> that the FFT in IDL 5.1.1 was faster than in IDL 5.1 on an other
>> PII-400 with Win95.
>
> I'm surprised the difference to Matlab 5.2 was so large.

Well, we have now done a sligthly more accurate test with the
following time results for the double precision FFT:

	1D	2D
IDL 5.1.0	9.9	2.58
IDL 5.1.1	5.9	1.91
Matlab 5.2	3.6	1.0

This means that Matlab is 1.9 times faster for 2D FFTs and
1.6 times faster for 1D FFTs than IDL 5.1.1 for this specific case.

The most surprising is the difference between IDL 5.1.0 and
IDL 5.1.1 - a performance gain of 1.68 for 1D FFTs.

There were less than 5% difference between Win-NT and Win95.

The numbers in the table is the execution time of (without the declaration)
1D: x = dcomplexarr(4096) & for i=0,999 do fft(x,-1,/DOUBLE)
2D: x = dcomplexarr(256,256) & for i=0,9 do fft(x,-1,/DOUBLE)

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by Liam Gumley on Wed, 16 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

FYI, Paul van Delst has written up a nice set of notes on how to use the
IDL FFT routine correctly. After all, if you're not getting the right
results, it doesn't matter how fast it is!

http://airs2.ssec.wisc.edu/~paulv/fft/fft_comparison.html

Cheers,
Liam.

Liam E. Gumley
Space Science and Engineering Center, UW-Madison
1225 W. Dayton St., Madison WI 53706, USA
Phone (608) 265-5358, Fax (608) 262-5974
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by steinhh on Wed, 16 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article <35ff8db4.0@d2o203.telia.com> roy.hansen@triad.no
(Roy E. Hansen) writes:
> We did a small comparison of the FFT performance in IDL 5.1.1
> compared with the Matlab 5.2 version for a PII-400 with Win-NT,
> and found that Matlab was approx 4 times faster. We also found
> that the FFT in IDL 5.1.1 was faster than in IDL 5.1 on an other
> PII-400 with Win95.

I'm surprised the difference to Matlab 5.2 was so large.

However, I know that the general performance of IDL went down

with the switch to C (rather than Fortran) as the source
language. I think this happened with the 3.6 -> 4.0 version
change. I did post some comparative timing estimates at
the time (not for the FFT though, as I recall).

>
> This raises a few questions:
>
> - Does there exist any optimized versions of IDL for the PII and
> PPro with W95 and Win-NT?
>
> - Does anybody know what the performance gain is using an optimized
> version compared to the standard version?
>
> - Is the IDL performance operating system dependent for the INTEL
> platform?

It may be that different compilers (thus different optimization
strategies) are used??

> - Are there any benchmarks of numerical performance for IDL
> compared to other software packages, like Matlab?

I would like to see some of those comparisons, too. Not just
for PCs, though.

> - If the FFTW (which is free) outperforms the native FFT in IDL,
> why don't RSI use that implementation? Is this a silly question?

Since FFTW is free under the Gnu Public License (GPL), RSI would
have to *give away* IDL under the GPL license in order to include
FFTW as part of IDL (I think). Not a good business practice! There
are two options, however:

1. RSI could produce a wrapper for FFTW and make it available
through their web site so the user could make a dynamically loadable
module (and the user would have to fetch the FFTW separately). AFAIK
this would mean that FFTW is not "sold as a part of IDL". Not really
a good solution, since programs using this module would crash on any
system that *didn't* have this "extra" installed.

2. The FFTW site mentions the possibility of non-free licenses. This
may be an idea to look into (after all, they are paying licences
for routines from Numerical Recipes).

However, I'm not so sure the *algorithm* of IDL's FFT is so bad,
I suspect (lacking) compiler optimizations to play a part here.
This is probably also the reason for the drop in performance

experienced in going from Fortran to C source code (Fortran code is easier to optimize - the compiler can make stronger assumptions on what is going on).

Anyway, RSI should keep an eye out to the competition wrt. the efficiency of basic mathematical routines in IDL vs other packages. Switching compilers (or maybe just turning on compiler switches :-) may be an option.

Regards,

Stein Vidar

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by stevenj on Thu, 17 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

David Kastrup <dak@mailhost.neuroinformatik.ruhr-uni-bochum.de> wrote:
> Karl Krieger <kak@ipp.mpg.de> writes:
>
>> BTW: I very much doubt if it's against the GPL to distribute code,
>> which refers to subroutine libraries under GPL as long as I do not
>> include these routines or a compiled binary.
>
> If the interface is unique to the GPL software, you are creating a
> derived work, as it is of no use without the GPL binary and is
> intended to link with it. The interface itself, however, is usually
> not considered copyrightable. So if you distribute a lousy
> implementation of fftw with the same interface along with your wrapper
> routines, one would have problems suing you in court.

That is an interesting point. Actually, you don't have to create a lousy implementation of FFTW. Versions of FFTW prior to 1.3 were not distributed under the GPL, but rather were free for non-commercial use. These versions used a subset of the current interface. So, as long as you stick to the 1.2 interfaces, I suppose you could argue that you are not creating a derived work of the GPL'ed versions. You still wouldn't be able to distribute something linked with the GPL'ed versions, though.

Cordially,
Steven G. Johnson

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by menakkis on Thu, 17 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

I hope that you folks don't mind me straying from IDL here, but there seems to be a fair amount of interest in getting FFTs to go as fast as possible. I saw something on the web the other day that might be of interest to IDL/WinNT users who have multiprocessor PCs...

Intel gives away a "maths kernel library" that includes 1D and 2D FFT routines. They claim that this stuff is nicely optimized for PPros / PIIs. They also state that the 2D FFT is multithreaded and will take advantage of a multiprocessor environment if you want it to, and - the tantalising part - that this will work even if your program (that calls their lib) is single-threaded. This means that you should be able to hook the Intel lib onto IDL, with a little "glue", and get some multiprocessor action going at last. (IDL is single-threaded.) The maths kernel also has some BLAS routines, a few of which are also multithreaded.

Check out: http://developer.intel.com/design/perftool/INDEX.HTM

Peter Mason

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by David Kastrup on Thu, 17 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Karl Krieger <kak@ipp.mpg.de> writes:

> BTW: I very much doubt if it's against the GPL to distribute code,
> which refers to subroutine libraries under GPL as long as I do not
> include these routines or a compiled binary.

If the interface is unique to the GPL software, you are creating a derived work, as it is of no use without the GPL binary and is intended to link with it. The interface itself, however, is usually not considered copyrightable. So if you distribute a lousy implementation of fftw with the same interface along with your wrapper routines, one would have problems suing you in court.

It then becomes the problem of the person dropping in the real fftw. As long as *he* does not distribute the compiled form of fftw... If he, however, distributes the stuff in an aggregation of useful subroutines, which happens to be on the same disk as the implementation including the lousy fftw.

Muddy waters.

--
David Kastrup Phone: +49-234-700-5570
Email: dak@neuroinformatik.ruhr-uni-bochum.de Fax: +49-234-709-4209
Institut fï¿½r Neuroinformatik, Universitï¿½tsstr. 150, 44780 Bochum, Germany

---

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by Karl Krieger on Thu, 17 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

On 17 Sep 1998, Stein Vidar Hagfors Haugan wrote:

>
> In article <stevenj-1609981832310001@wetelectron.mit.edu>
> stevenj@alum.mit.edu (Steven G. Johnson) writes:
>
> ...
>
> This may be splitting hairs, but I could imagine myself writing
> such wrappers for FFTW (and indeed it appears that Karl Krieger has
> already done so) or other GPL-covered libraries.
> ...

If anybody is interested in the FFTW wrapper routines, feel free
to contact me by email. At present they handle only a subset of the
FFTW routines (1d and 2d) and documentation is rudimentary at best.
I have some example IDL progs though. Makefiles can be provided for
SUN-Solaris (cc and gcc) and for M$ Visual C++ 5.0.

BTW: I very much doubt if it's against the GPL to distribute code, which
refers to subroutine libraries under GPL as long as I do not include
these routines or a compiled binary.

Karl
--
Max-Planck-Institute for Plasma Physics
Boltzmannstr.2, 85740 Garching, Germany Email: krieger@ipp.mpg.de

---

Subject: Re: IDL performance and FFTs (was: call external speed)
Posted by steinhh on Thu, 17 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article <stevenj-1609981832310001@wetelectron.mit.edu>
stevenj@alum.mit.edu (Steven G. Johnson) writes:

[...]

> [steinhh@ulrik.uio.no (Stein Vidar Hagfors Haugan) wrote:]
>> 1. RSI could produce a wrapper for FFTW and make it available
>> through their web site so the user could make a dynamically loadable
>> module (and the user would have to fetch the FFTW separately). AFAIK
>> this would mean that FFTW is not "sold as a part of IDL".
>
> Nope, you can't get around the GPL in this way. To quote R. Stallman of
> GNU, "A GPL-covered plug-in that is designed to be combined with [only] a
> non-free master program is a form of combined work, and a violation of the
> GPL." (The original authors can make an exception allowing their code to
> be used in such a plugin, but no one else can do so.)
>
> (Otherwise, the GPL would essentially devolve to the LGPL--you could link
> any GPL'ed code you wanted into a non-free program just by making it a
> "plugin." For more info, do a search on Dejanews for: ~a
> (rms@santafe.edu) & ~g (gnu.misc.discuss) & "Plug-ins")
>
> You can make such a plugin for your own use, but you can't distribute it.
> (In any case, you are probably right in that such a plugin wouldn't be
> widely useful unless it came with IDL.)

I see the point that RSI cannot supply an operational plugin. Such a
plugin would appear as much a part of IDL as e.g. the jpeg/hdf/cdf
support routines, which are actually situated in dynamically loaded
modules (plugins), which *could* have been written by third parties.

What I was suggesting was (the unlikely scenario) that RSI
would write a short piece of C code that takes care of the type
checking, extracting the array sizes etc, before calling
FFTW functions (that are not supplied by RSI).

The user (or his/her system manager) would have to fetch the FFTW
code from the original web site (http://theory.lcs.mit.edu/~fftw/),
and make sure it got linked together with the piece of code supplied
by RSI.

I.e., RSI would only give instructions on, *how* to make a plugin
out of the GPL-covered code. The user would make the plugin,
by combining the instructions and the wrapper code.

This may be splitting hairs, but I could imagine myself writing
such wrappers for FFTW (and indeed it appears that Karl Krieger has
already done so) or other GPL-covered libraries.

Would I really be breaking the GPL licence by giving away *only*
this:

1. C file containing calls to GPL-covered libraries

2. Instructions on how to get the GPL-covered library
3. Makefile that links my C file and the GPL-covered
   library into a plugin for IDL.

If so, I've nearly done a bummer with my regular expression DLM, since it's possible to use the GPL-covered regex package to provide the regcomp, regexec, regerror and regfree routines.

(And thanks for the tip on Fortran/C optimizations)

Regards,

Stein Vidar
(*No* expert on GPL!)
