Subject: Problems with the IDL TIME_TEST
Posted by rsmith on Thu, 11 Jul 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Rick Shafer, my officemate here at NASA Goddard, was looking over the IDL benchmarks rather carefully recently, trying to make sure everything was kosher...and there seems to be a ham&cheese sandwich in there somewhere.

The problem is in the floating-point multiplies...here's the code, cut-n-pasted from the library, for the byte-multiplies and the floating:

```
....
a=replicate(2b,512,512)
reset
for i=1,10 do b=a*2b
timer,'Mult 512 by 512 byte by constant and store, 10 times'
for i=1,100 do c = shift(b,10,10)
timer,'Shift 512 by 512 byte and store, 100 times'
for i=1,50 do b=a+3b
timer,'Add constant to 512 x 512 byte array and store, 50 times'
for i=1,30 do b=a+b
timer,'Add two 512 by 512 byte images and store, 30 times'

a = float(a)
reset
for i=1,30 do b=a*2b
timer,'Mult 512 by 512 floating by constant and store, 30 times'
for i=1,30 do c = shift(b,10,10) ; should be c = b + 2.0
timer,'Add constant to 512 x 512 floating and store, 30 times'
for i=1,30 do b=a+b
timer,'Add two 512 by 512 floating images and store, 30 times'
....
```

It looks to us as though RSI just copied the code, which is not correct, of course, for floating points. Changing the code slightly, as noted above, results in a change in the benchmark from 0.43 to 0.75, on my Dec Alpha, for an ~40% slowdown, in a fairly important benchmark.

Randy Smith (& Rick Shafer)
randall.smith@gsfc.nasa.gov
rick.shafer@gsfc.nasa.gov

Subject: Re: Problems with the IDL TIME_TEST
Posted by David Ritscher on Thu, 18 Jul 1996 07:00:00 GMT

> I run this test on my PC, which has an Intel i486: I got the same numbers
> whether the multiplier was 2.000000 or 3.141592.
> However, I got different times when the matrix was filled with !PI rather
> than 4.000000 as in the original program. More funny: Multiplying !pi
> with 4.000000 takes a lot more time than multiplying with !pi
> ** on this CPU ***

Yes, !pi takes much longer to load than 3.14, as the following
tests indicate. I guess there is some extra system overhead involved,
just like there is in function calls, etc.

I repeated a series of tests 15 times, similar to the following:

```
t=systime(1)
for i = 0L,200000 do z=3
times(0,now) = systime(1)-t
```

And took the median of the result, for each of the following assignment
statements:

```
z=3
z=3.14
z=3.14D
z=mypi
z=!pi
z=float(3)
z=!dpi
z=sin(3.14)
z=abs(3.14)
z=!pi*!pi
z=!pi*3.1
z=3.1*3.1
```

I tested them on both IDL Version 4.0.1 and PV-WAVE CL Version 6.01,
on an HP 715/64 workstation. Here are the timing results
(seconds per 200000 iterations):

Operation	PV-Wave	IDL
z=3	2.23400	0.880200
z=3.14	2.22800	0.881200
z=3.14D	2.23000	0.878900
z=mypi	1.93000	0.895000
z=!pi	5.10500	2.12260
z=float(3)	7.07500	2.76390
z=!dpi	5.49100	2.14240
z=sin(3.14)	7.49700	3.35770
z=abs(3.14)	6.45100	2.69320
z=!pi*!pi	9.82000	4.38980
z=!pi*3.1	7.07700	3.21930
z=3.1*3.1	4.20000	1.90350

As can be seen, using !pi instead of 3.14 or a variable to which the system variable !pi were assigned requires over twice the execution time. It is somewhat similar to times involving function calls.

If the above operations are done with vectors instead of loops, i.e.,

```
zz = fltarr(/nozero, 200000)
z=3.14
t=systime(1)
zz(*) = z
print,systime(1)-t
```

Then it no longer matters whether !pi or 3.14159 is used, since the variable is only looked up once. The time for all of the different operations are essentially identical within each system:

PV-Wave	IDL
0.35 s	0.70 s

No, I have no explanation for why IDL is twice as fast with the loop approach and PV-Wave is twice as fast with the vector approach. I tried these both from the command line and as a compiled .pro file and it made no difference in the times.

It's worth taking a look at these results graphically - just grab the following with your mouse, and pass it on to an IDL or PV-Wave window:

```
idl = [ 0.8802,  0.8812,  0.8789,  0.8950,  2.1226,  2.7639,  2.1424,
3.3577,  2.6932,  4.3898,  3.2193,  1.9035]
wave = [ 2.2340,  2.2280,  2.2300,  1.9300,  5.1050,  7.0750,  5.4910,
7.4970,  6.4510,  9.8200,  7.0770,  4.2000]
labels = ['z=3', 'z=3.14', 'z=3.14D', 'z=mypi', 'z=!!pi', 'z=float(3)',
'z=!dpi', 'z=sin(3.14)', 'z=abs(3.14)', 'z=!!pi*!!pi', 'z=!!pi*3.1',
'z=3.1*3.1']
n_tests = 12
!x.title = 'Test number'
!y.title = 'Time (s) / 200,000 iterations of each test
!p.multi = [0, 1, 2]
plot, wave
!p.title = 'Assignment tests using PV-Wave'
for i=0, n_tests-1 do xyouts, i, wave(i), labels(i)
!p.title = 'Assignment tests using IDL'
plot, idl
for i=0, n_tests-1 do xyouts, i, idl(i), labels(i)
```

--

David Ritscher
Raum 47.2.401
Zentralinstitut fuer Biomedizinische Technik
Albert-Einstein-Allee 47
Universitaet Ulm
D-89069 ULM
Germany

Tel: ++49 (731) 502 5313
Fax: ++49 (731) 502 5315
internet: david.ritscher@zibmt.uni-ulm.de