Subject: Re: Problems with the IDL TIME_TEST
Posted by hahn on Fri, 12 Jul 1996 07:00:00 GMT
View Forum Message <> Reply to Message

rsmith@zardoz.gsfc.nasa.gov (Randall Smith) wrote:

> 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.

Let me add two remarks here: IDL uses the time-of-day for measuring.
1) For reliable results you need an empty machine, no other users.
2) The time of day clock isn't updated very often on a couple of systems,
   maybe only 60 times per second (no, it is nor synchronized to the power
   supply), giving a resolution of 16 msec. Thus, if one of the tests (shifting
   a matrix) only needs 0.04 sec, you should run the test several times. With
   the speed of todays CPUs (Pentium Pro @ 200 MHz) we actually need
   time_tes3.pro rather than time_tes2.pro ....
   ... And watch paging!!

> 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'

[snip]

> a = float(a)
> reset
> for i=1,30 do b=a*2b

As the type of a if float, IDL will convert 2B to float before multiplying
the matrix. However, this is done 30 time because of the loop.

> timer,'Mult 512 by 512 floating by constant and store, 30 times'

(snip]

> 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.

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 ***

I know from other benchmarks that CPU behave differently depending on the values of the operands. This is because a floating multiply is made of shifting the mantissae until the exponents are the same and then multiplying the mantissae. Multiplication is skipped when the multiplier is 0 or 1. So the time for a multiplication highly depents on the number of binary zeroes and ones of the mantissa!!

The 40% slowdown you observerd on a DEC Alpha is very common!

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

Hope this helps
Norbert Hahn
