Subject: x*x versus x^2
Posted by Conor on Wed, 09 Jul 2008 16:32:57 GMT
View Forum Message <> Reply to Message

So I've been looking at execution time for various algorithms, and I found this interesting result:

```
bigarr = fltarr(1000,1000)

t1 = systime(/seconds)
t = bigarr^2.0
t2 = systime(/seconds)
t = bigarr*bigarr
t3 = systime(/seconds)

print,t2-t1
print,t3-t2
```

IDL prints:

```
    0.024163008
    0.010262012
```

Apparently multiplying an array by itself is twice as fast as using the carat operator! Anyone know why this is? Is it a memory issue or something?

Subject: Re: x*x versus x^2
Posted by Sven Geier on Sat, 12 Jul 2008 19:30:22 GMT
View Forum Message <> Reply to Message

Conor wrote:

```
>  So I've been looking at execution time for various algorithms, and I
>  found this interesting result:
>
>  bigarr = fltarr(1000,1000)
>
>  t1 = systime(/seconds)
>  t = bigarr^2.0
>  t2 = systime(/seconds)
>  t = bigarr*bigarr
>  t3 = systime(/seconds)
>
>  print,t2-t1
>  print,t3-t2
```

>
>
> IDL prints:
>
>   0.024163008
>   0.010262012
>
> Apparently multiplying an array by itself is twice as fast as using
> the carat operator! Anyone know why this is? Is it a memory issue or
> something?

What version of IDL and what OS are we talking about here? I tried to reproduce this (IDL 6.3 Win32/x86) and I get wildly varying results every time I hit F5:

```
IDL> .GO
    0.016000032
    0.014999866
IDL> .GO
     0.00000000
    0.016000032
IDL> .GO
    0.016000032
     0.00000000
```

The help on SYSTIME() tels me that on windows this simply doesn't seem to have the accuracy to measure quantities in the small-number-of-milliseconds range...

When I increase the array sizes by a factor of 10 in each dimension I get the following:

```
IDL> .GO
     0.45400000
     0.90599990
IDL> .GO
     0.62500000
     0.75000000
IDL> .GO
     0.64000010
     0.76600003
IDL> .GO
     0.62500000
     0.75000000
```

Not exactly repeatable.

However this is all based on multiplication of zero with zero -- to what

degree is that optimized under the hood? If I replace the first line in
your script

```
  bigarr = fltarr(1000,1000)
```

with

```
  bigarr = randomn(seed,10000,10000)
```

I get a completely different picture:

```
IDL> .GO
       5.1099999
      0.75000000
IDL> .GO
       5.2340000
      0.76600003
IDL> .GO
       5.1559999
      0.78100014
IDL> .GO
       5.1410000
      0.76600003
```

Replacing the float exponent "2.0" with a plain "2" turns this into:

```
IDL> .GO
      0.65700006
      0.75000000
IDL> .GO
      0.64100003
      0.75000000
IDL> .GO
      0.65700006
      0.76500010
IDL> .GO
      0.64100003
      0.76500010
```

Finally:
replacing bigarr^2 and bigarr*bigarr
    with bigarr*2 and bigarr+bigarr

Gives me the following:

```
IDL> .GO
      0.53200006
      0.76500010
```

```
IDL> .GO
      0.53099990
      0.75000000
IDL> .GO
      0.54700017
      0.75000000
IDL> .GO
      0.51500010
      0.76599979
IDL> .GO
      0.50000000
      0.78200006
IDL> .GO
      0.48399997
      0.76600003
```

Just to add some data points...

--
http://www.sgeier.net
My real email address does not contain any "Z"s.
