Subject: Re: x*x versus x^2
Posted by dzhang49 on Sat, 12 Jul 2008 16:25:54 GMT
View Forum Message <> Reply to Message

On Jul 9, 11:10 am, Conor <cmanc...@gmail.com> wrote:
> On Jul 9, 12:57 pm, Bruce Bowler <bbow...@bigelow.org> wrote:
>
>
>
>> On Wed, 09 Jul 2008 09:43:27 -0700, Conor wrote:
>>> On Jul 9, 12:32 pm, Conor <cmanc...@gmail.com> 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?
>
>>> This also holds true for array's smaller than the multi-threading
>>> minimum size, so it isn't because multi-threading is being used in one
>>> case but not the other...
>
>> Digging into the deep dark recesses of my brain...
>
>> exponentiation with a real exponent generally uses the log function to do
>> it's thing. *some* language implementations are smart enough that if the
>> exponent is an integer, they decompose the exponentiation into
>> multiplication.
>
>> It might be worth trying your experiment with t=bigarr^2 and see how the
>> results change.
>

>> Bruce
>
> Interesting... I tried your suggestion and got this result:
>
> 0.018048048
> 0.010533094
>
> So it is still slower, but the difference is smaller. A calculation
> like this is rarely the bottleneck for speed in a program, so I
> probably won't worry about it too much, but it is an interesting fact
> to be aware of...

Actually, if you increase the dimension of the array, the result will
be reverse, here it is:

```
pro test_speed
  bigarr = fltarr(10000,10000)
  t1 = systime(/seconds)
  t = bigarr^2.0
  t2 = systime(/seconds)
  t = bigarr*bigarr
  t3 = systime(/seconds)

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

and the results are :
0.68420601
0.83076620

So if you run a larger number, the ^2 will be faster