Subject: Re: strange behaviour of bytscl by large arrays
Posted by manodeep@gmail.com on Tue, 24 Apr 2012 17:10:30 GMT
View Forum Message <> Reply to Message

On Apr 24, 10:40 am, Chris Torrence <gorth...@gmail.com> wrote:
> On Tuesday, April 24, 2012 8:50:46 AM UTC-6, alx wrote:
>> On 23 avr, 22:22, Chris Torrence <gorth...@gmail.com> wrote:
>>> On Monday, April 23, 2012 10:14:21 AM UTC-6, fawltyl...@gmail.com wrote:
>
>>>> I think IDL's FINDGEN() implementation is wrong: it uses a float counter instead of an integer one. The following test shows the difference:
>
>>>> pro test
>>>> cpu, tpool_nthreads=1
>>>> n=10l^8
>>>> nn=n-1
>>>> a1=findgen(n) ; real FINDGEN()
>>>> a2=fltarr(n)
>>>> count=0.0
>>>> for j=0l, nn do a2[j]=count++ ; IDL's implementation
>>>> a3=fltarr(n)
>>>> count=0ll
>>>> for j=0l, nn do a3[j]=count++ ; better implementation
>>>> print, a1[nn], a2[nn], a3[nn], format='(3F15.3)'
>>>> end
>
>>>> (Multithreading must be disabled because the starting values for the threads are calculated as an integer. So the result of FINDGEN() depends on the number of your CPU cores, too :-)
>
>>>> regards,
>>>> Lajos
>
>>> Well, wrong is perhaps too strong of a word. The real word is "fast". I just did a test where I changed the internal implementation of FINDGEN to use an integer counter. The "float" counter is 4 times faster than using an integer counter and converting it to floats.
>
>>> However, perhaps we could look at the size of the input array, and switch to using the slower integer counter if it was absolutely necessary. I'll give it a thought.
>
>>> Thanks for reporting this!
>
>>> Cheers,
>>> Chris
>>> Exelis VIS
>
>> It is risky to write a statement like "findgen(n)" while n is larger
>> than the inverse of the floating point precision (given in IDL by
>> long(1/machar().eps)). This is true in any programming language. It is

>> mathematically incorrect to assume that such a "findgen" will behave
>> as a "lindgen".
>> IDL is not "wrong" here, but rather clever. Is'nt it ?
>> alx.
>
> Okay, alx has convinced me to not change anything. Try the following:
>
> IDL> print, 16777216 + findgen(10), format='(f25.0)'
> 16777216.
> 16777216.
> 16777218.
> 16777220.
> 16777220.
> 16777220.
> 16777222.
> 16777224.
> 16777224.
> 16777224.
>
> So even if you did the computation using long64's, as soon as you convert them back to floats, you are going to get "jumps" in the findgen because of the loss of precision. I suppose you could argue that this might be better than having the findgen get "stuck" on the number 16777216, but I think the speed of findgen is more important.
>
> Thanks.
>
> -Chris
> Exelis VIS

It looks like IDL is actually behaving "correctly" - consider the
following C code:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int main(void)
{
  float  fx = powf(2.0,24);
  double dx = pow(2.0,24);
  const double d_one = 1.0;
  const float f_one = 1.0;
  const int N=10;
  for(int i=0;i<N;i++){
    fprintf(stderr,"i=%5d (float) x = %25.10f  (double) x = %25.10lf
\n",i,fx+(i*f_one),dx+(i*d_one));
```

```
  }

  return EXIT_SUCCESS;
}
----------------------------------------
```

The output from the code is:

```
i=    0 (float) x =       16777216.0000000000  (double) x =
16777216.0000000000
i=    1 (float) x =       16777216.0000000000  (double) x =
16777217.0000000000
i=    2 (float) x =       16777218.0000000000  (double) x =
16777218.0000000000
i=    3 (float) x =       16777220.0000000000  (double) x =
16777219.0000000000
i=    4 (float) x =       16777220.0000000000  (double) x =
16777220.0000000000
i=    5 (float) x =       16777220.0000000000  (double) x =
16777221.0000000000
i=    6 (float) x =       16777222.0000000000  (double) x =
16777222.0000000000
i=    7 (float) x =       16777224.0000000000  (double) x =
16777223.0000000000
i=    8 (float) x =       16777224.0000000000  (double) x =
16777224.0000000000
i=    9 (float) x =       16777224.0000000000  (double) x =
16777225.0000000000
```

which is exactly what IDL prints out for findgen.

Cheers,
Manodeep