Subject: Re: Why is MEAN so slow?
Posted by wlandsman on Wed, 19 Jan 2011 01:36:02 GMT

View Forum Message <> Reply to Message

On Tuesday, January 18, 2011 7:52:45 PM UTC-5, Matthew Francis wrote:

> I've been using the PROFILER to track down why some code was a bit
> slow and found that it was spending most of its time in the MEAN
> function (and then within that, in the MOMENT function called by
> MEAN).
>

In IDL 8.0, MEAN no longer calls MOMENT but does the calculation itself using code similar to your MEAN_QUICK.   (This is part of the upgrade that also added a DIMENSION keyword.)

But I don't see anything inefficient about the earlier code which called MOMENT().   In fact, I find the same processing times whether using MEAN_QUICK, the pre-8.0 MEAN(), or the V8.0 MEAN().   (My test consisted of taking the mean of a 5000 x 5000 randomn array with selected values set to NAN.)

--Wayne

Subject: Re: Why is MEAN so slow?
Posted by Matt Francis on Wed, 19 Jan 2011 02:14:04 GMT

View Forum Message <> Reply to Message

Interesting. I've only tested this for 1 dimensional arrays on IDL
7.1, not for matrices (the application that I was trying to speed up
only used MEAN on 1D arrays).

Here is how I am comparing the two with a test code:

```
-----------------------
pro test_mean_speed

  ; Test whether the inbuilt MEAN function is slower than MEAN_QUICK

  n = 100000

  data = randomu(seed,100000)

  indx = where(data GT 0.9,count)

  data[indx] = !values.f_nan

  for i=0,10000 do begin
    ;foo=mean_quick(data,/nan)
    foo=mean(data,/nan)
  endfor
```

```
end
-------------------------------
```

I ran this once using MEAN and once using MEAN_QUICK. Here are the full results from PROFILER

For MEAN:

```
Module         Type  Count     Only(s)   Avg.(s)     Time(s)   Avg.
(s)
ABS            (S)   10001    5.040137  0.000504    5.040137
0.000504
ARG_PRESENT     (S)   10001    0.005722  0.000001    0.005722
0.000001
FINITE         (S)   10001    3.328896  0.000333    3.328896
0.000333
KEYWORD_SET     (S)   30003    0.014247  0.000000    0.014247
0.000000
MEAN           (U)   10001    0.025580  0.000003   61.421387
0.006142
MOMENT         (U)   20002   32.259783  0.001613  108.917069
0.005445
N_ELEMENTS      (S)   20002    0.008905  0.000000    0.008905
0.000000
ON_ERROR       (S)   30003    0.012295  0.000000    0.012295
0.000000
RANDOMU        (S)       1    0.001750  0.001750    0.001750
0.001750
SIZE           (S)   10001    0.011161  0.000001    0.011161
0.000001
SQRT           (S)   10001    0.007367  0.000001    0.007367
0.000001
TEST_MEAN_SPEED (U)       1    0.006986  0.006986   61.430769
61.430769
TOTAL          (S)   60006   18.553728  0.000309   18.553728
0.000309
WHERE          (S)   10002    2.154211  0.000215    2.154211
0.000215
```

For MEAN_QUICK:

```
Module         Type  Count     Only(s)   Avg.(s)     Time(s)   Avg.
(s)
FINITE         (S)   10001    3.247247  0.000325    3.247247
0.000325
KEYWORD_SET     (S)   10001    0.004842  0.000000    0.004842
0.000000
MEAN_QUICK      (U)   10001    1.432719  0.000143    9.902029
```

```
0.000990
RANDOMU        (S)      1    0.001732  0.001732    0.001732
0.001732
TEST_MEAN_SPEED (U)      1    0.011449  0.011449    9.915830
9.915830
TOTAL         (S)   10001    3.133684  0.000313    3.133684
0.000313
WHERE         (S)   10002    2.084156  0.000208    2.084156
0.000208
```

According to PROFILER (as well as the obvious difference in how long they ran for), MEAN_QUICK is much faster, for this specific problem.

Subject: Re: Why is MEAN so slow?
Posted by wlandsman on Wed, 19 Jan 2011 19:08:05 GMT

View Forum Message <> Reply to Message

On Tuesday, January 18, 2011 9:14:04 PM UTC-5, Matthew Francis wrote:

> Interesting. I've only tested this for 1 dimensional arrays on IDL
> 7.1, not for matrices (the application that I was trying to speed up
> only used MEAN on 1D arrays).

It turns out that there is a bug in the moment.pro function in IDL 7.1 (but not in 7.0 or before, or in 8.0).   There is a MAXMOMENT keyword that is supposed to tell moment.pro not to calculate higher order moments, so if one only wants the mean, then one sets MAXMOMENT = 1.   But if one also supplies /NaN, then MOMENT calls itself recursively after removing the NaN values. But due to a typo, the MAXMOMENT keyword was not being transmitted, and the program defaults to MAXMOMENT = 4.   So the reason mean.pro was 5 times slower than your program is that all the higher order moments were being calculated.   (MOMENT underwent a major rewrite for 8.0 and no longer calls itself recursively.)

Another mystery was why, in IDL 8.0, the IDL mean.pro function is almost twice as fast as your mean_quick.pro function for your example.   The reason is that it does not use the WHERE function -- you want to know how many NaN values there are, but you don't care where they are. Here is how one would modify mean_quick.pro to not use WHERE   --Wayne

```
function mean_quick8,data,nan=nan,double=double

  if keyword_set(nan) then begin
    count =total(~finite(data,/Nan),/integer)
    if count EQ 0 then return,  $
       keyword_set(double) ? !values.D_nan : !values.f_nan
       return,    total(data,double=double,/nan)/count
  endif else begin
    return,total(data,double=double)/n_elements(data)
  endelse
```

end

Subject: Re: Why is MEAN so slow?
Posted by Foldy Lajos on Wed, 19 Jan 2011 19:49:54 GMT
View Forum Message <> Reply to Message

On Wed, 19 Jan 2011, wlandsman wrote:

```
>  On Tuesday, January 18, 2011 9:14:04 PM UTC-5, Matthew Francis wrote:
>>  Interesting. I've only tested this for 1 dimensional arrays on IDL
>>  7.1, not for matrices (the application that I was trying to speed up
>>  only used MEAN on 1D arrays).
>
```

> It turns out that there is a bug in the moment.pro function in IDL 7.1 (but not in 7.0 or before, or in 8.0).   There is a MAXMOMENT keyword that is supposed to tell moment.pro not to calculate higher order moments, so if one only wants the mean, then one sets MAXMOMENT = 1.   But if one also supplies /NaN, then MOMENT calls itself recursively after removing the NaN values. But due to a typo, the MAXMOMENT keyword was not being transmitted, and the program defaults to MAXMOMENT = 4.   So the reason mean.pro was 5 times slower than your program is that all the higher order moments were being calculated.   (MOMENT underwent a major rewrite for 8.0 and no longer calls itself recursively.)
>
> Another mystery was why, in IDL 8.0, the IDL mean.pro function is almost twice as fast as your mean_quick.pro function for your example.   The reason is that it does not use the WHERE function -- you want to know how many NaN values there are, but you don't care where they are. Here is how one would modify mean_quick.pro to not use WHERE   --Wayne

```
>
>  function mean_quick8,data,nan=nan,double=double
>
>   if keyword_set(nan) then begin
>     count =total(~finite(data,/Nan),/integer)
>     if count EQ 0 then return,  $
>        keyword_set(double) ? !values.D_nan : !values.f_nan
>        return,    total(data,double=double,/nan)/count
>   endif else begin
>     return,total(data,double=double)/n_elements(data)
>   endelse
>
>  end
>
```

Changing

```
        count =total(~finite(data,/Nan),/integer)
```

to

```
        count =n_elements(data)-total(finite(data,/Nan),/integer)
```

makes the counting even faster.

regards,
Lajos

Subject: Re: Why is MEAN so slow?
Posted by Matt Francis on Wed, 19 Jan 2011 22:06:38 GMT
View Forum Message <> Reply to Message

Thanks a lot to you both. Sounds like I just got unlucky with the 7.1 bug. The shiny new extra extra fast version of MEAN is making the data processing much faster!