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

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
```