Subject: An optimisation question
Posted by Matt Francis on Tue, 27 Mar 2012 02:33:08 GMT
View Forum Message <> Reply to Message

Hi all, I am trying to optimise some IDL code and I'm getting results
I don't expect.

I have a 3 dimensional array which describes the values of a set of
basis function on a 2D regular grid. The first element in the index of
the basis function, then the lat and lon. I then have a pair of lat
lon vectors which are a list of the indices of the points in the 2D
grid that a set of data points sit at. I want to find the value of the
basis functions for these data points.

Now, first for some background. In 1D if i had a basis function and an
index vector then clearly:

```
>  values = basis[ix]
```

is much faster than

```
>  for i=0,N_ELEMENTS(values)-1 do values[i] = basis[ix[i]]
```

Now, for 2D we can do a similar thing:

```
>  values = basis[ilat,ilon]
```

the RHS will return a 1D arrary of length equal to the length of ilat
and ilon.

Now, for 3D something different happens. The first element of my array
is the index of the basis function, so I tried to do:

```
>  values = basis[index,ilat,ilon]
```

But, in this case, the RHS returns an array of dimensions [1, N, N]
where N is the length of ilat/ilon

This can do done in a loop:

```
>  for i=0,N do values[i] = basis[index,ilat[i],ilon[i]]
```

But loops are slow (and this is indeed a real bottleneck in my code).
I attempted to make a loopless version:

```
>  values = (REFORM(basis[index,*,*]))[ilat,ilon]
```

In this case the RHS returns an array of the correct length, however
for the size of basis array I have, the second version turns out to be
literally hundreds of times slower (according to PROFILER). I though
this must be because of REFORM, but the time spent in the REFORM
function is relatively small. I can't work out why the second version
is slow?

Here is a test code snippet isolating this conundrum. In this case the
REFORM approach is only 5 times slower, so not as bad as my example,
but why is it slow at all? Surely the FOR loop is not the optimal
approach here!

```
pro test_foo_1,basis,lat,lon,ret
  ret = (reform(basis[0,*,*]))[lat,lon]
end

pro test_foo_2,basis,lat,lon,ret
  for i=0,9999 do $
    ret[i]=basis[0,lat[i],lon[i]]
end

pro test_foo
  Basis = fltarr(10,1000,1000)
  lat = fltarr(10000)
  lon = fltarr(10000)

  ret = fltarr(10000)

  test_foo_1,basis,lat,lon,ret
  test_foo_2,basis,lat,lon,ret

end
```

Profiler result:

```
Module         Type  Count    Only(s)   Avg.(s)     Time(s)   Avg.
(s)
FLTARR         (S)      4    0.010160  0.002540    0.010160
0.002540
PROFILER       (S)      2    2.413247  1.206624    2.413247
1.206624
REFORM         (S)      1    0.000004  0.000004    0.000004
0.000004
TEST_FOO       (U)      1    0.000011  0.000011    0.022829
0.022829
TEST_FOO_1     (U)      1    0.010315  0.010315    0.010319
0.010319
TEST_FOO_2     (U)      1    0.002339  0.002339    0.002339
```

0.002339