Subject: Re: An optimisation question
Posted by Yngvar Larsen on Tue, 27 Mar 2012 09:35:37 GMT
View Forum Message <> Reply to Message

On Tuesday, 27 March 2012 04:33:08 UTC+2, Bogdanovist wrote:

> 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

Hm. Interesting. I've noticed before that one-liner 1D loops are quite fast in IDL. I also cannot explain why your TEST_FOO_1 is so slow since REFORM seems not to be the culprit. However, you can do it much more efficiently like this:

```
pro test_foo_1,basis,lat,lon,ret,slice
  ret[0] = basis[slice+lonarr(n_elements(lat)),lat,lon]
end
```

Profiler report:

```
IDL> profiler & profiler, /system
IDL> for n=0,99 do test_foo
IDL> profiler, /report
Module          Type  Count     Only(s)   Avg.(s)     Time(s)   Avg.(s)
FLTARR           (S)    400    1.646489  0.004116    1.646489  0.004116
LONARR           (S)    100    0.001611  0.000016    0.001611  0.000016
```

```
N_ELEMENTS      (S)     100    0.000101  0.000001    0.000101  0.000001
PROFILER        (S)       2    0.000193  0.000097    0.000193  0.000097
TEST_FOO        (U)     100    0.001783  0.000018    1.917648  0.019176
TEST_FOO_1      (U)     100    0.015138  0.000151    0.016837  0.000168
TEST_FOO_2      (U)     100    0.252603  0.002526    0.252603  0.002526
```

PS: In your test programs, LAT and LON are used as index arrays, but declared as floating point arrays. Not that it matters here though.

--
Yngvar

Subject: Re: An optimisation question
Posted by Matt Francis on Tue, 27 Mar 2012 21:26:58 GMT
View Forum Message <> Reply to Message

Thanks for that, much faster!

Do you comprehend why this is so much faster? I really hate not understanding what IDL is doing 'under the hood' with this type of thing. I have a lot of code doing some similar things that I need to optimise and it would be nice to have a better understanding, rather than just using trial and error. What are the good rules of thumb that are in operation here?

By the way, the Lat/Lon arrays are integers in the real code, don't know why I made them floats for this example!

Subject: Re: An optimisation question
Posted by David Fanning on Tue, 27 Mar 2012 22:00:38 GMT
View Forum Message <> Reply to Message

Matt Francis writes:

>> 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?

I suspect it is those pesky asterisks giving you trouble

in your loop.

http://www.idlcoyote.com/code_tips/asterisk.html
http://www.idlcoyote.com/misc_tips/submemory.html

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: An optimisation question
Posted by Matt Francis on Tue, 27 Mar 2012 22:45:32 GMT
View Forum Message <> Reply to Message

Okay, just probing my original approach further to understand what is going on and I'm going completely insane. Have a look at this test code:

```
function do_3d,arr,lat,lon,indx
  temp = reform(arr[indx,*,*])
  return,temp[lat,lon]
end

function do_2d_first,arr,lat,lon
  return,arr[lat,lon]
end

function do_2d_second,arr,lat,lon
  return,arr[lat,lon]
end

pro crazy_idl
  lat = intarr(10000)
  lon = intarr(10000)

  arr3d = fltarr(10,1000,1000)
  arr2d = fltarr(1000,1000)

  for i=0,100 do begin
    res3d = do_3d(arr3d,lat,lon,0)
```

```
    res2d = do_2d_first(arr2d,lat,lon)
    res2d = do_2d_second(reform(arr3d[0,*,*]),lat,lon)
  endfor
end
```

The '3d' version first uses REFORM to obtain a 2d matrix and then does the same thing as the '2d' version. The second call to the 2d version does the REFORM command before sending the array to the subroutine. All three approaches are essentially the same, apart from some minor overhead coming from using REFORM. Well, no. Apparently these are all very different! Check out the profiler report:

Module	Type	Count	Only(s)	Avg.(s)	Time(s)	Avg.(s)
CRAZY_IDL	(U)	1	0.967564	0.967564	1.963462	1.963462
DO_2D_FIRST	(U)	101	0.003964	0.000039	0.003964	0.000039
DO_2D_SECOND	(U)	101	0.004870	0.000048	0.004870	0.000048
DO_3D	(U)	101	0.973172	0.009635	0.973531	0.009639
FLTARR	(S)	2	0.013167	0.006583	0.013167	0.006583
INTARR	(S)	2	0.000012	0.000006	0.000012	0.000006
PROFILER	(S)	2	1.007490	0.503745	1.007490	0.503745
REFORM	(S)	202	0.000711	0.000004	0.000711	0.000004

What the hell?? One of either me or IDL is doing something completely screwy and frankly I don't care which it is, I just want to understand what is going on. I guess the other possibility is that the profiler is getting this completely wrong a misreporting the times in some weird way?

Subject: Re: An optimisation question
Posted by Matt Francis on Tue, 27 Mar 2012 22:54:18 GMT
View Forum Message <> Reply to Message

I think I see now (previous post cross posted with DF). Reading David's links I see now that the culprit is the arr[indx,*,*] part (those pesky asterisks..). I had assumed that was trivial but on further inspection of the profiler report that operation alone it taking up the bulk of the runtime!

Thanks for all the help.
