Subject: Arrays suck. Loops rock.
Posted by Benjamin Hornberger on Fri, 04 Mar 2005 18:22:22 GMT
View Forum Message <> Reply to Message

If you're sick of discussions about loops, better skip to the next
message. If not, read on.

I'm disappointed. I've been told loops suck and arrays rock in IDL. I've
read the dimension juggling tutorial
(http://www.dfanning.com/tips/rebin_magic.html) and was striving hard to
eliminate all loops from my code. Until today. When suddenly my code
with two FOR loops ran twice as fast as loopless. One loop was right in
between.

The scientific case: image_data is a 3-d array, a stack of "channels" of
the same image [n_fast_pixels, n_slow_pixels, n_data] (take fast and
slow as x and y, if you want). A "display" is a linear combination of
channels. linear_combinations is a 2-d array [n_data, n_displays]
holding the coefficients for n_displays interesting displays. The
entries will typically be -1, 0, or 1.

The code below calculates "displays", a 3-d array [n_fast_pixels,
n_slow_pixels, n_displays] holding all these interesting displays in one
array. I can do it loopless or with one or two loops. In the array
calculation, I even packed everything into one expression, making the
code quite unreadable, to make sure I don't lose time by making
unnecessary copies of arrays. With two loops it's twice as fast as
without, with one loop it's right in between.

Yes, I also heard that FOR loops are not always evil. Still, it would be
great if somebody could shed some light on why in this particular case
loops are faster. Do the rebin / reform steps take a lot of time? If I
have to execute that code many times, do I gain by saving and reusing
the result of the rebin / reform step?

BTW, I also hit the point when I didn't have enough memory to hold the
4-d intermediate array. Then arrays *really* suck. The task manager had
to come rescue me.

This is all on Win XP, IBM Thinkpad T40 (Pentium M, 1.6 GHz, 512 MB).

```
IDL> test_loop,fltarr(300,300,10),fltarr(10,20),disps,nloops=0
took       0.82099986 sec
IDL> test_loop,fltarr(300,300,10),fltarr(10,20),disps,nloops=1
took       0.58000016 sec
IDL> test_loop,fltarr(300,300,10),fltarr(10,20),disps,nloops=2
took       0.47000003 sec
```

```
PRO test_loop, image_data, linear_combinations, $
            displays, nloops=nloops

  on_error, 2

  time = systime(/sec)

  IF n_elements(nloops) EQ 0 THEN nloops = 0

  svec = size(image_data)
  n_fast_pixels = svec[1]
  n_slow_pixels = svec[2]
  n_data = svec[3]

  IF (size(linear_combinations))[1] NE n_data THEN $
    message, 'dimensions wrong'

  n_displays = (size(linear_combinations))[2]

  IF nloops EQ 0 THEN BEGIN

     ;; we build a 4-dimensional array [n_fast_pixels, n_slow_pixels,
     ;; n_data, n_displays], and in the end TOTAL over the
     ;; n_data dimension.

     displays = total(rebin(reform(linear_combinations, 1, 1, $
                                   n_data, n_displays), $
                            n_fast_pixels, n_slow_pixels, $
                            n_data, n_displays) * $
                      rebin(image_data, n_fast_pixels, n_slow_pixels, $
                            n_data, n_displays), $
                      3)

  ENDIF ELSE BEGIN

     displays = fltarr(n_fast_pixels, n_slow_pixels, n_displays)

     FOR i=0, n_displays-1 DO BEGIN

        IF nloops EQ 1 THEN BEGIN

           displays[*, *, i] = $
             total(rebin(reform(linear_combinations[*, i], $
                                1, 1, n_data, /over), $
                         n_fast_pixels, n_slow_pixels, n_data) * $
                   image_data, $
                   3)
        ENDIF ELSE IF nloops EQ 2 THEN BEGIN

           FOR j=0, n_data-1 DO BEGIN
              displays[*, *, i] = displays[*, *, i] + $
                             linear_combinations[j, i] * $
                             (image_data)[*, *, j]
           ENDFOR

        ENDIF
     ENDFOR

  ENDELSE
  displays = reform(displays, $
               n_fast_pixels, $
               n_slow_pixels, $
               n_displays, /overwrite)

  print, 'took ', systime(/sec)-time, ' sec'

END
```