Subject: Re: For-loop vs. Dimensional Juggling relative performance
Posted by cgguido on Wed, 10 Feb 2010 02:54:08 GMT
View Forum Message <> Reply to Message

On Feb 8, 10:26 pm, Gray <graylikethecо...@gmail.com> wrote:
> Hi folks,
>
> I recently wrote my own version of SRCOR from the NASA Astrolib. Just
> as a reminder, the program takes two lists of 2D coordinates and finds
> matches where the distance is less than some cutoff. SRCOR uses a for-
> loop to step through the first list, comparing the distance of each
> coordinate-pair from every point in the second list. My version uses
> matrix multiplication and dimensional juggling to avoid the for-loop.
>
> For n1 = 2143 and n2 = 2115, SRCOR is faster (0.16 seconds to my 0.53
> on my macbook); however, for n1 = 25 and n2 = 26, mine is faster
> (1.8e-4 seconds to 4.2e-4). Is there any way to predict what kind of
> list sizes will be faster with each method, without making some random
> data and using brute force?
>
> The relevant code is:
>
> SRCOR (dcr2 is the cutoff, option eq 2 ignores the cutoff) -->
>
> FOR i=0L,n1-1 DO BEGIN
> xx = x1[i] & yy = y1[i]
> d2=(xx-x2)^2+(yy-y2)^2
> dmch=min(d2,m)
> IF (option eq 2) or (dmch le dcr2) THEN BEGIN
> ind1[nmch] = i
> ind2[nmch] = m
> nmch = nmch+1
> ENDIF
> ENDFOR
>
> My code -->
>
> lkupx = rebin(indgen(n1),n1,n2) ;make index lookup
> tables, so as not to
> lkupy = rebin(transpose(indgen(n2)),n1,n2) ;worry about confusing
> 1D vs 2D
> ;use matrix multiplication and dim. juggling to fast compute
> sqrt((x2-x1)^2+(y2-y1)^2)
> dists =
> sqrt(rebin(x1^2.+y1^2,n1,n2)+rebin(transpose(x2^2.+y2^2),n1, n2)-2*(x1#x2+y1#y2))
> min_x = min(dists,xmatch,dimension=2) ;find the minima in both
> directions...
> min_y = min(dists,ymatch,dimension=1) ;this is given in 1D indices

>   xm = lkupy[xmatch] ;convert to 2D indices
>   ym = lkupx[ymatch]
>   ;remove elements w/ distance greater than max_dist, and where the
> two lists don't match
>   nomatch_x = where(ym[xm] ne indgen(n1) or min_x gt max_dist, nmx)
>   if (nmx gt 0) then xm[nomatch_x] = -1
>   nomatch_y = where(xm[ym] ne indgen(n2) or min_y gt max_dist, nmy)
>   if (nmy gt 0) then ym[nomatch_y] = -1
>
> Thanks!!
> --Gray (first time poster)

Gray, have you tried the inbuilt DISTANCE_MEASURE ? I'd be curious to know if it's any faster.

--Gianguido
