Subject: Re: Optimizing loops
Posted by Russell[1] on Wed, 09 Dec 2015 16:29:29 GMT
View Forum Message <> Reply to Message

Continuing on this thread...

It looks like you're trying to match two sets of coordinates. Well... unfortunately that is a very slow calculation, and I suspect even if you optimize your code like Sergey suggests you won't see a major improvement. The only way you'll get this faster is if you think way out side of the box.

You can use two sets of histograms to chunk your catalogs into sets of the maximum distance you expect to find a match (looks like your thr14). Then you basically turn one big loop into 2 smaller loops...First loop over chunks that have entries (which is by definition smaller than the entire dataset). Then loop over other objects which also fall in that bin. Again, also very small. David Fanning describes it here:

http://www.idlcoyote.com/code_tips/matchlists.html

I coded this up to work in Cartesian space and it is 10-1000 times faster than what you have. Well, i did this long before I read your post, but my code was pretty much exactly what Sergey suggested. This code will be a pain to write because it's a lot of juggling of reverse_indices and what not, but when it runs, it will be MUCH faster. The cases where this algorithm seem to break down is when you have a VERY sparse catalog, such that there is only one entry per chunk cell (but even then it's still a few factors faster than the naive approach).

-Russell

PS, the code is deeply embedded in an object and would be a pain to excise and post here. But it's similar in philosophy to what David has, and other pieces form such IDL luminaries as JD Smith, W Landsman, and C Markwardt.

On Wednesday, December 2, 2015 at 2:04:33 PM UTC-5, sam.t...@gmail.com wrote:
> Hello all! I am working with a large number of satellite data files. The files are quite big, but I've never had this much trouble working with them before, even while doing similar processing.
>
> The basic program structure that's the slowest is below; it also just dominates the processing power as it enters this loop. I need to do this for each file I process.
>
> ; we want to calculate the percent of each satellite pixel
> ; covered by land
> land_perc = FLTARR(N_ELEMENTS(field1))
> FOR j = 0, N_ELEMENTS(field1)-1 DO BEGIN
> dist = (((land_lon - sat_lon[j])*COSD(land_lat))^2. $
> + (land_lat - sat_lat[j])^2.)^0.5*111.12

```
>
>         ind14 = WHERE(dist LE 14.)
>         land14 = land_mask(ind14)
>         landy = WHERE(land14 EQ 0, landy_cnt)
>         land_perc[j] = FLOAT(landy_cnt)/FLOAT(N_ELEMENTS(land14))*100
>      ENDFOR
>
>  If anyone has optimization suggestions, please let me know! Thanks :)
```
