Subject: Optimizing loops
Posted by sam.tushaus on Wed, 02 Dec 2015 19:04:28 GMT
View Forum Message <> Reply to Message

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 :)

Subject: Re: Optimizing loops
Posted by Sergey Anfinogentov on Thu, 03 Dec 2015 10:10:19 GMT
View Forum Message <> Reply to Message

```
> 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 :)
```

There is no problem with loops here because the code inside the loop is vectorised.
The code is slow because of two things
1)The computational complexity is of the order of N^2. For large N it will be slow even if coded in C or Fortran.
2)In present form, COSD function is called N times. I guess, land_lat is an array. Sum functionsFunctions like SIN, COS, ALOG are computationally very heavy and take a lot of time to be called.
3) Square root "^0.5" is computed many times inside loop. It is also very computationally heavy operation.

You can try the following steps to improve the performance

(1) If land_lat is a large array, you can improve the performance by calculating COSD only one time before the loop:
cosd_land_lat = COSD(land_lat)
Then you should use "cosd_land_lat" instead of "COSD(land_lat)" inside the loop.
(2) don't calulate square root of inside the loop. Compute dist^2 instead of dist and compare it with the modified trhreshold.

The optimised code may look like this

```
   land_perc = FLTARR(N_ELEMENTS(field1))
   cosd_land_lat = COSD(land_lat)
   thr14 = (14./111.12)^2
   FOR j = 0, N_ELEMENTS(field1)-1 DO BEGIN
      dist2 = (((land_lon - sat_lon[j])*cosd_land_lat)^2. $
           + (land_lat - sat_lat[j])^2.)

      ind14 = WHERE(dist2 LE thr14)
      land14 = land_mask(ind14)
      landy = WHERE(land14 EQ 0, landy_cnt)
      land_perc[j] = FLOAT(landy_cnt)/FLOAT(N_ELEMENTS(land14))*100
   ENDFOR
```

Hope, it will help

Subject: Re: Optimizing loops
Posted by Phillip Bitzer on Tue, 08 Dec 2015 16:28:48 GMT
View Forum Message <> Reply to Message

On Wednesday, December 2, 2015 at 1:04:33 PM UTC-6, sam.t...@gmail.com wrote:

> If anyone has optimization suggestions, please let me know! Thanks :)

In addition to the suggestions that Sergey provided, let's add a couple more:

```
>         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
```

For the last line, no need to "float" both numbers. Also, you might consider either a) dropping the *100 and do that at the end or b) reordering the operations so that you don't need to call FLOAT at all. Consider:

```
IDL> 1/2  ;we know this is not the number we are looking for
IDL> 1/FLOAT(2) ;closer, but we'll have to multiply the final array by 100
IDL> 100.0 * 1/2 ;this works, but only if the "100.0" comes first
```

Also, you don't need to count the number of elements in land14 - you already have that information. It's the same size as ind14, and you can get that size using the count argument.

Further, x*x is faster than x^2...although the cost/benefit depends on the size of the arrays.

Finally, there *is* a way to further vectorize your code. But, whether it's worth it depends on the size of the land/sat arrays. There's some black magic using rebins, pound signs, etc.

Subject: Re: Optimizing loops
Posted by MarioIncandenza on Tue, 08 Dec 2015 19:05:07 GMT

View Forum Message <> Reply to Message

On Wednesday, December 2, 2015 at 11:04:33 AM UTC-8, 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 :)

So you have these 2D arrays: LAND_LON, LAND_LAT, LAND_MASK
And these 2D (or 1D) arrays (different dims): SAT_LON, SAT_LAT
You want to calculate the fraction of land in a 14km radius from each satellite pixel. This can be done analytically using these steps:
1) Calculate binary land/water us BLAND=(LAND_MASK NE 0);
2) Project binary land mask BLAND to 1km (or 250m or whatever) rectangular grid (MAP_PROJ_INIT(), MAP_PROJ_FORWARD(),INTERPOLATE(INTERPOL());
3) Calculate fractional "land within 14km":
 KERNEL=sqrt((rebin(findgen(29),29,29)-14)^2+(rebin(findgen(1 ,29),29,29)-14)^2) ge 14; binary +/-14km
FLAND14_1KM = CONVOL(FLOAT(BLAND_1KM),KERNEL,/EDGE_TRUNCATE)
4) Project satellite LON/LAT onto 1km grid (MAP_PROJ_FORWARD);
5) Sample FLAND14_1KM at SAT_LON/SAT_LAT:
SAT_IX_1KM=INTERPOL(FINDGEN(N_ELEMENTS(X_1KM)),X_1KM,SAT_X_1 KM)
 SAT_IY_1KM=INTERPOL(FINDGEN(N_ELEMENTS(Y_1KM)),Y_1KM,SAT_Y_1 KM)
FLAND14_SAT = FLAND14_1KM[SAT_IX_1KM,SAT_IY_1KM]

This will give you a numerically robust solution in a single step for your entire granule.
NOTE 1) This is memory-intensive and if you really need a global grid at 250 meters you might not have enough memory for it.
NOTE 2) If you use a global grid, the answers near the poles and dateline will be suspect.

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 :)