Subject: Optimizing code for faster calculation
Posted by Kenneth D on Thu, 13 Mar 2014 06:33:31 GMT
View Forum Message <> Reply to Message

I've been looking at this block of code now for... ever.

I've been editing a program created by my Adviser to reduce run time wherever possible. So far I've reduced the run time by nearly half, and I'm trying to juice any performance I can get from absolutely anywhere. My final project will use an array roughly 17,000 by 17,000. And I have to iterate through the program at least 17,000*10 times. If I'm lucky it won't take a month to process my data-sets now. This code is about all I have left to work with:

```
 exceed_subs = where(min_rmse GT rmse_threshold, counter)
 if counter GT 0 then modeled_class(exceed_subs) = "unmodeled"
```

min_rmse is an array Float[200], such as [0.347272, 0.312437, 0.360164,...]
rmse_threshold = 0.025
modeled_class is an array String[200], such as ["soil","quag","soil","grass",...]

The code find the locations where min_rmse is greater than a threshold value, and replaces those index locations in the string array (modeled_class) with "unmodeled".

This may well be the most efficient way to do this (this code will run a minimum of 17,000 times) but a look at the histograms page at Exelis:
http://www.exelisvis.com/docs/HISTOGRAM.html

shows:
For example, make the histogram of array A:
H = HISTOGRAM(A, REVERSE_INDICES = R)
;Set all elements of A that are in the ith bin of H to 0.
IF R[i] NE R[i+1] THEN A[R[R[I] : R[i+1]-1]] = 0

;The above is usually more efficient than the following:
bini = WHERE(A EQ i, count)
IF count NE 0 THEN A[bini] = 0

Which looks so similar to what I'm trying to do. I tried to implement this with no luck (maybe because strings?). Is there anything else I can do? That is, besides taking out iterations, they simply must be there to do what I need.

Subject: Re: Optimizing code for faster calculation
Posted by Helder Marchetto on Thu, 13 Mar 2014 07:55:15 GMT
View Forum Message <> Reply to Message

On Thursday, March 13, 2014 7:33:31 AM UTC+1, Kenneth D wrote:
> I've been looking at this block of code now for... ever.
>

>
>
> I've been editing a program created by my Adviser to reduce run time wherever possible. So far I've reduced the run time by nearly half, and I'm trying to juice any performance I can get from absolutely anywhere. My final project will use an array roughly 17,000 by 17,000. And I have to iterate through the program at least 17,000*10 times. If I'm lucky it won't take a month to process my data-sets now. This code is about all I have left to work with:
>
>
>
> exceed_subs = where(min_rmse GT rmse_threshold, counter)
>
> if counter GT 0 then modeled_class(exceed_subs) = "unmodeled"
>
>
>
> min_rmse is an array Float[200], such as [0.347272, 0.312437, 0.360164,...]
>
> rmse_threshold = 0.025
>
> modeled_class is an array String[200], such as ["soil","quag","soil","grass",...]
>
>
>
> The code find the locations where min_rmse is greater than a threshold value, and replaces those index locations in the string array (modeled_class) with "unmodeled".
>
>
>
> This may well be the most efficient way to do this (this code will run a minimum of 17,000 times) but a look at the histograms page at Exelis:
>
> http://www.exelisvis.com/docs/HISTOGRAM.html
>
>
>
> shows:
>
> For example, make the histogram of array A:
>
> H = HISTOGRAM(A, REVERSE_INDICES = R)
>
> ;Set all elements of A that are in the ith bin of H to 0.
>
> IF R[i] NE R[i+1] THEN A[R[R[I] : R[i+1]-1]] = 0
>
>
>

> ;The above is usually more efficient than the following:
>
> bini = WHERE(A EQ i, count)
>
> IF count NE 0 THEN A[bini] = 0
>
>
>
> Which looks so similar to what I'm trying to do. I tried to implement this with no luck (maybe because strings?). Is there anything else I can do? That is, besides taking out iterations, they simply must be there to do what I need.

Hi,
I think you should have a look at
http://www.idlcoyote.com/tips/histogram_tutorial.html
You will find the information you need in there.

That said, my guess is that you will need to set the proper binsize in your histogram command. Depending on the type of values you have, you might try using binsize = 0.025, but I don't have time to check if that is a good option.
Hope it helps.
Cheers,
Helder

Subject: Re: Optimizing code for faster calculation
Posted by David Fanning on Thu, 13 Mar 2014 12:28:12 GMT
View Forum Message <> Reply to Message

Kenneth D writes:

> I've been looking at this block of code now for... ever.

I think you are in good shape here. I'd go worry about something else.
Chances are you only going to have to run this program once. :-)

Cheers,

David

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

Subject: Re: Optimizing code for faster calculation
Posted by Heinz Stege on Thu, 13 Mar 2014 13:34:10 GMT
View Forum Message <> Reply to Message

On Wed, 12 Mar 2014 23:33:31 -0700 (PDT), Kenneth D wrote:

> Is there anything else I can do? That is, besides taking out iterations, they simply must be there to do what I need.

Since the array min_rmse has only 200 elements, I doubt that it is possible to make this part of the code significantly faster.

Otherwise for me it seems, that "modeled_class" beeing a string array is more the bottleneck than the where function. It may help to change the code to something like the following. Start with the lines which can be executed somewhere in the header of your code:

```
modeled_class=lonarr(200)
temp=["soil","quag","grass",...,"unmodeled"]  ; unique values only!
modeled_class_names=temp[sort(temp)]
unmodeled=value_locate(modeled_class_names,"unmodeled")
```

Then write the respective indicees into the long integer array modeled_class.

This way the lines whithin the iteration can be changed to

```
exceed_subs = where(min_rmse GT rmse_threshold, counter)
if counter GT 0 then modeled_class(exceed_subs) = unmodeled
```

Are such changes possible with your code?

Cheers, Heinz

Subject: Re: Optimizing code for faster calculation
Posted by chris_torrence@NOSPAM on Thu, 13 Mar 2014 13:46:02 GMT
View Forum Message <> Reply to Message

> This way the lines whithin the iteration can be changed to
>
>
>
> exceed_subs = where(min_rmse GT rmse_threshold, counter)
>
> if counter GT 0 then modeled_class(exceed_subs) = unmodeled
>
>

>
> Are such changes possible with your code?
>
>
>
> Cheers, Heinz

I would agree with Heinz. Strings in general are slow. Heck, if you only have 200, I would just use a byte array for the modeled_class index values. You can then convert them at the very end (outside of the loop!) to the string values.

Also, not sure what version of IDL you're using (I'm a bit worried that you are using parentheses for indexing, which is never a good sign...), but if you have IDL 8.0 or higher, you could eliminate the "if counter" check by using the NULL keyword to where:

modeled_class[WHERE(min_rmse gt rmse_threshold, /NULL)] = unmodeled

This will be a "no-op" if none of the values match, and will do the right thing if some of the values match.

Cheers,
Chris
ExelisVIS

Subject: Re: Optimizing code for faster calculation
Posted by Kenneth D on Fri, 14 Mar 2014 00:46:28 GMT
View Forum Message <> Reply to Message

Thank you so much!

Converting my strings to index values and processing like this:

modeled_class[where(min_rmse GT rmse_threshold, /NULL)] = unmodeled_index

Totally made a difference! Now the biggest ding on my program is the built in function: min()

min_rmse = min(rmse_subset, min_subs, dimension=1)

I'm going to see if a histogram of this might be faster or something...

Additional Info:
My test case is a 200x200 matrix. My real world case is a 17000x17000 matrix
I'm using IDL 8.2.#, I tried my code on 8.3 at our university and the index (i) gave me an error, so I changed those to [i]. I blame it on switching between Python and IDL.
