Subject: FOR loops and efficiency
Posted by Rachel on Thu, 21 May 2009 16:04:43 GMT
View Forum Message <> Reply to Message

It was impressed upon me sometime or another that the use of FOR loops in IDL applications tends to be an inefficient solution for doing many tasks, yet sometimes I have difficulty finding a reasonable alternative to the FOR loop. I was wondering if anyone could give me advice on the following example code.

I am trying to make a function that takes arrays of parameters and then generates a mathematical model. In the following example I use gaussian curves, but generally I would want to expand an implementation to other mathematical functions (gaussians are just easy for this example).
So basically I can accomplish what I want to do using something like the following:

```
x = findgen(2000)*0.1 + 900.0
y = fltarr(2000)+1.0

lam0 = findgen(10)*50.0 + 900.0
depth = findgen(10)/10.0
width = findgen(10)

for i = 0,n_elements(lam0)-1 do y = y *(1.0 - depth[i]*exp(-(x-width
[i])^2/2.0/width[i]))
```

I was thinking about how one might accomplish the same things without a for loop and I came up with the following... problem being that for large arrays of lam0 this is actually more inefficient (I'm assuming because of the use of extraordinarily large arrays)

```
n = n_elements(x)
nlines = n_elements(lam0)
y = product(1.0 - rebin(transpose(depth),n,nlines)*exp(-(rebin
 (x,n,nlines)-rebin(transpose(lam0),n,nlines))^2/2.0/rebin(tr anspose
(width),n,nlines)),2)
```

any advise?

Thanks!
Josh

Subject: Re: FOR loops and efficiency
Posted by Craig Markwardt on Sat, 23 May 2009 20:24:41 GMT

On May 22, 2:24 pm, Christopher Thom <ct...@oddjob.uchicago.edu>
wrote:
> Quoth Craig Markwardt:
>
>> A FOR loop will only be slow(er) when the time spent executing the
>> loop overhead is much more than the time spent doing the computations
>> in one loop iteration. A simple test would be to execute a dummy
>> loop:
>> NMAX = 100000L
>> FOR I = 0L, NMAX do begin & dummy = 1
>> Keep raising the value of NMAX until the execute time of the loop is
>> perceptible. Don't bother trying to optimize loops smaller than this.
>
>> In your case, you are only doing ten iterations, and each iteration
>> does a lot of work, so you won't gain by removing the loop.
>
> I've heard this description about FOR loops a lot, but one general
> question I've never been able to answer is, "how do i know when my loops
> are doing enough work?". How do I know when my loop overhead is a large
> fraction of the time spent on an iteration?
>
> I guess the real underlying question here is recognising when to optimise,
> and when to simply move on to more important things. Does anyone have any
> rules of thumb to help guide this recognition?

I still stand by my rule of thumb. The problem with FOR loops is the
amount of time spent doing loop overhead stuff. If you run your loop
but *take all the calculations out*, and the total execution time is
not perceptible, then you probably won't gain by optimizing/
vectorizing.

Craig

Subject: Re: FOR loops and efficiency
Posted by JDS on Tue, 26 May 2009 21:51:06 GMT

> I still stand by my rule of thumb. The problem with FOR loops is the
> amount of time spent doing loop overhead stuff. If you run your loop
> but *take all the calculations out*, and the total execution time is
> not perceptible, then you probably won't gain by optimizing/
> vectorizing.

I find that analysis lacking for a few reasons. Consider this
example:

```
IDL> a=randomu(sd,10000000L)
IDL> t=systime(1) & b=total(a,/CUMULATIVE) & print,systime(1)-t
     0.066554070
IDL> t=systime(1) & for i=0L,10000000L-2 do a[i+1]+=a[i] &
print,systime(1)-t
       2.2091250
IDL> print,array_equal(a,b)
   1
```

The non-loop version was >30x faster. So by your rule of thumb, our simple loop with its small calculation must be totally dominated by loop overhead. Let's check that:

```
IDL> t=systime(1) & for i=0L,10000000L-2 do begin & end & print,systime
(1)-t
      0.12700295
```

Not so much. The loop overhead contributed only roughly 5% of the total computation time, and yet the simple call to TOTAL blew it out of the water. Why?

Loop overhead is one reason to avoid FOR loops with high iteration count, but it is by no means not the *only* reason. Element by element memory access is less efficient, multiple processor cores cannot typically be used in the loop statement, and no doubt many other code/data fetch efficiency factors come into play. That's why there is no simple answer to "when is a FOR loop a problem". The only real answer is "when non-FOR loop methods can be found, are faster, and you need the extra speed." My rough rule of thumb: if you are accessing many hundreds or thousands of elements in each iteration, you are probably not being impacted by efficiency issues, though you will lose potential multicore speedup.

Sometimes you encounter a problem which just requires a loop. IDL *really needs* the ability to easily call out to C code which can be automatically compiled (ideally by a tool distributed with IDL itself) into efficient form, and which can be integrated directly into the session. MATLAB has had most of this capability for some time. There are hacks using MAKE_DLL, but they are tricky, depend on user compiler installation, and are consequently rarely used.

JD

Subject: Re: FOR loops and efficiency
Posted by Craig Markwardt on Thu, 28 May 2009 05:38:20 GMT

On May 26, 5:51 pm, JDS <jdtsmith.nos...@yahoo.com> wrote:
>> I still stand by my rule of thumb. The problem with FOR loops is the
>> amount of time spent doing loop overhead stuff. If you run your loop
>> but *take all the calculations out*, and the total execution time is
>> not perceptible, then you probably won't gain by optimizing/
>> vectorizing.
>
> I find that analysis lacking for a few reasons. Consider this
> example:
...
> IDL> t=systime(1) & for i=0L,10000000L-2 do a[i+1]+=a[i] &print,systime(1)-t
...
> IDL> t=systime(1) & for i=0L,10000000L-2 do begin & end & print,systime(1)-t
> 0.12700295
...
> Loop overhead is one reason to avoid FOR loops with high iteration
> count, but it is by no means not the *only* reason. ...

I agree with everything you said. I still stand by my guideline as
rule of thumb to know when optimization is important. Note that the
rule of thumb didn't involve trying to compare the execution time of
an empty loop and a full loop. :-)

By the way, if you put a simple dummy statement like this,

t=systime(1) & for i=0L,10000000L-2 do begin & dummy = 0 & end &
print,systime(1)-t

Then the execution time is more like 0.5 seconds. While I agree that
this is not the same as 2.2 seconds, it is definitely more
comparable.

Craig

Subject: Re: FOR loops and efficiency
Posted by JDS on Fri, 29 May 2009 20:39:16 GMT

On May 28, 1:38 am, Craig Markwardt <craig.markwa...@gmail.com> wrote:
> On May 26, 5:51 pm, JDS <jdtsmith.nos...@yahoo.com> wrote:
>
>
>
>>> I still stand by my rule of thumb. The problem with FOR loops is the
>>> amount of time spent doing loop overhead stuff. If you run your loop

>>> but *take all the calculations out*, and the total execution time is
>>> not perceptible, then you probably won't gain by optimizing/
>>> vectorizing.
>
>> I find that analysis lacking for a few reasons. Consider this
>> example:
> ...
>> IDL> t=systime(1) & for i=0L,10000000L-2 do a[i+1]+=a[i] &print,systime(1)-t
> ...
>> IDL> t=systime(1) & for i=0L,10000000L-2 do begin & end & print,systime(1)-t
>> 0.12700295
> ...
>> Loop overhead is one reason to avoid FOR loops with high iteration
>> count, but it is by no means not the *only* reason. ...
>
> I agree with everything you said. I still stand by my guideline as
> rule of thumb to know when optimization is important. Note that the
> rule of thumb didn't involve trying to compare the execution time of
> an empty loop and a full loop. :-)
>
> By the way, if you put a simple dummy statement like this,
>
> t=systime(1) & for i=0L,10000000L-2 do begin & dummy = 0 & end &
> print,systime(1)-t
>
> Then the execution time is more like 0.5 seconds. While I agree that
> this is not the same as 2.2 seconds, it is definitely more
> comparable.
>
> Craig

Right. Now that I read your rule of thumb more carefully, I see your point is really "keep it to a small number of iterations." The problem is, if this is in a function which itself (perhaps later) gets called millions of times, it will be of no solace that each function call performs only 100 iterations. But your rule of thumb is actually useful for all types of performance optimization. Should I optimize? Only if it takes too long.

JD

Subject: Re: For loop
Posted by penteado on Mon, 26 Oct 2009 15:01:03 GMT
View Forum Message <> Reply to Message

On Oct 26, 11:54 am, Hassan <hkhav...@gmail.com> wrote:
> Hi,

>
> I wonder if I can use 'For... Do Begin....End For ' in the command
> line. I can use this in a pro code and it works well but in command
> line it gives me errors. I just can use 'For... Do' in command line
> but it's not suitable if I have several commands into the loop. Could
> you pleasse help me about that.
>
> Cheers,
> Hassan.

In the command line (as well as in a batch file), you need to provide
the entire loop "in one line". That is, you need to use & to put all
the lines of your loop into one, as in:

for i=0,1 do begin & print,i & print,i & endfor

But that is only proper for something short, like two or three short
statements in the loop. Do not abuse it. If you need to write a long
block it should not be done in this way, it should be in a function or
procedure.

Subject: Re: For loop
Posted by Hassan on Mon, 26 Oct 2009 15:45:28 GMT
View Forum Message <> Reply to Message

On Oct 26, 3:01 pm, pp <pp.pente...@gmail.com> wrote:
> On Oct 26, 11:54 am, Hassan <hkhav...@gmail.com> wrote:
>
>> Hi,
>
>> I wonder if I can use 'For... Do Begin....End For ' in the command
>> line. I can use this in a pro code and it works well but in command
>> line it gives me errors. I just can use 'For... Do' in command line
>> but it's not suitable if I have several commands into the loop. Could
>> you pleasse help me about that.
>
>> Cheers,
>> Hassan.
>
> In the command line (as well as in a batch file), you need to provide
> the entire loop "in one line". That is, you need to use & to put all
> the lines of your loop into one, as in:
>
> for i=0,1 do begin & print,i & print,i & endfor
>
> But that is only proper for something short, like two or three short
> statements in the loop. Do not abuse it. If you need to write a long

> block it should not be done in this way, it should be in a function or
> procedure.

Thanks a lot for your help. Hassan

Subject: Re: For loop
Posted by Michael Galloy on Mon, 26 Oct 2009 16:47:50 GMT
View Forum Message <> Reply to Message

Hassan wrote:
> Hi,
>
> I wonder if I can use 'For... Do Begin....End For ' in the command
> line. I can use this in a pro code and it works well but in command
> line it gives me errors. I just can use 'For... Do' in command line
> but it's not suitable if I have several commands into the loop. Could
> you pleasse help me about that.
>
> Cheers,
> Hassan.

Here's another technique for command line usage:

```
IDL> .run
- for i = 0, 1 do begin
-   print, i
-   print, i
- endfor
- end
% Compiled module: $MAIN$.
       0
       0
       1
       1
```

The same warnings apply: this is good for a line or two, if you need more I would put the code in a file.

Mike
--
www.michaelgalloy.com
Research Mathematician
Tech-X Corporation

Subject: Re: For loop

Posted by Hassan on Tue, 27 Oct 2009 08:16:52 GMT
View Forum Message <> Reply to Message

Thanks a lot Mike.