Subject: Re: Philosophy of for loops
Posted by Craig Markwardt on Mon, 28 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

tclement@ucsd.edu (Todd Clements) writes:
>
> So what defines a slow loop? Is it having a bunch of accesses to
> sub-elements of arrays? Is it just having a bunch of statments? I suppose
> I could do some tests of my own, and I have a little, but it's much more
> fun to hear what you all have to say on the subject. I wouldn't have seen
> any IDL-ku if I just kept my thoughts to myself!

A good question. I think there are two parts to it, and you are on the right track. Theory comes first, then some practical solutions.

IDL is a scripted language, so almost by definition very little optimization can be done. Sure, the IDL code is "compiled," but what that really means is that the IDL statements are converted to some intermediate form, sometimes called bytecode. These bytecodes represent a higher level of abstraction and portability compared to true machine code, but with that comes the burden that each bytecode requires many many machine code operations to implement.

Therein lies the rub. Even a simple FOR loop, there's always going to be a lot of "overhead" or extra processing. Darn. This is the price to have IDL be so general purpose. Another thing to keep in mind is that basically *everything* gets re-evaluated each iteration. When you compare the following two examples:

```
for i = 0L, N-1 do x(i) = f(x(i).tag)    ; vs.
x = f(x.tag)
```

you should realize that in the first, the values of I, X(I), X(I).TAG and F(X(I).TAG) are re-evaluated every iteration, because IDL is dynamic. In the second example the operands are evaluated once.

What this really boils down to is, *reduce* the number of iterations, and do more per iteration. This will minimize the useless processing compared to the "real" processing.

You can do more per iteration with vectorization. The vectorized operations *are* heavily optimized for speed, but you need to know how to take advantage of them. That is left as an exercise to the reader :-), but obviously it's often non-trivial, and sometimes impossible.

Let's see what this boils down to.

Q: When is the number of iterations, N, too many?

A: When the execution time of the empty loop becomes perceptible:
FOR I = 0L, N-1 DO BEGIN & END
(this is about a million for my faster machine)

Q: How much vectorization should I do?

A: Generally it is sufficient to only vectorize the *innermost* loop.
If you have an image with rows and columns, and you can vectorize
the operation at the row-level, you should be safe. This is often
described as *chunking* or *banding*.

Hope this helps!
Craig

for when=you, must do $
many, many, iterates, $
vectorize (a=chunk)

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: Philosophy of for loops
Posted by John-David T. Smith on Tue, 29 Aug 2000 05:14:37 GMT
View Forum Message <> Reply to Message

Todd Clements wrote:
>
> Hello, again, everyone!
>
> I was just wondering what the general concensus of the "IDL Expert
> Programmers" was on the use of for loops. When I first learned IDL, I
> remember getting from someone or somewhere the mantra "for loops are evil"
> because they take up so much time. Of course, as I learn more and watch
> what goes on in this group, it seems like "for loops are sometimes evil"
> would be a better mantra. The question then becomes, when do they become
> evil?
>
> In response to my thread on summing diagonal elements, Craig said that for
> loops aren't always bad if you can do a lot at once, and his code proves
> that you can have some fast loops.
>
> So what defines a slow loop? Is it having a bunch of accesses to
> sub-elements of arrays? Is it just having a bunch of statments? I suppose
> I could do some tests of my own, and I have a little, but it's much more

> fun to hear what you all have to say on the subject. I wouldn't have seen
> any IDL-ku if I just kept my thoughts to myself!

Despite what many supposedly learned men of the cloth will say to the contrary, for loops are without question the embodiment of pure evil. It is only the demonic influences to which these unfortunate souls have yielded which speak, and we must take their lamentable and unholy descent as frightening examples of the ever-burgeoning power of the dark, unoptimized side.

The only for loops you should ever sully your hands with are those involving the occasional pointer or object arrays, and then only disdainfully and with a careful eye to preserving your hard fought IDL sanctity and virtue, lest you slip into the evaluation of individual rows or columns at a time, or even, may the gods curse it, *nested* for loops. And yes, the dark side can be seductive with its adding of matrix diagonals in a few simple lines, its promise to iterate "just a few times".

But simply compare the snake-like slitheriness and so-called "readability" of:

```
**************************************
tt = fltarr(nx+ny-1)
ll = lindgen(nx>ny)
for i = 0, ny-1 do tt(i)      = total((a(0+ll,i-ll))(0:i<(nx-1)))
for i = 1, nx-1 do tt(i+ny-1) = total((a(i+ll,ny-1-ll))(0:(nx-1-i)<(ny-1)))
**************************************
```

to the most properly considered and well-balanced:

```
**************************************
tt=total(a[(((dy=((di=lindgen(((n=nx<ny)),nx+ny-1)))/n))*(nx gt ny?1:nx)+ $
           (nx gt ny?1:-1)*((dx=di mod n))*(nx-1))>0<(nx*ny-1)]* $
         (dy ge dx AND (dy-dx) lt nx>ny),1)
**************************************
```

which would stand in your code like a pillar of sunlight, shining firmly through the firmament. Oh yes, the wretched demon seducters of the nether-codeland will pass before your eyes many beautiful visions, but a closer look will reveal a false beauty, marred to the point of hateful grotesqueness. But despair not! If you take upon yourself a solemn and unquavering vow to reveal and combat these foul and insidious creatures on each journey you undertake, down every darkened lane of inquest; if you stand firmly and resolutely in steadfast devotion to the good Vector; if you read aloud daily from the holy book of Histogram; than shall you triumph over IDL with all the powers of the right and just at your aid.

Your humble guide,

JD

Subject: Re: Philosophy of for loops
Posted by marc schellens[1] on Tue, 29 Aug 2000 06:43:26 GMT
View Forum Message <> Reply to Message

Todd Clements wrote:
>
> Hello, again, everyone!
>
> I was just wondering what the general concensus of the "IDL Expert
> Programmers" was on the use of for loops. When I first learned IDL, I
> remember getting from someone or somewhere the mantra "for loops are evil"
> because they take up so much time. Of course, as I learn more and watch
> what goes on in this group, it seems like "for loops are sometimes evil"
> would be a better mantra. The question then becomes, when do they become
> evil?
>
> In response to my thread on summing diagonal elements, Craig said that for
> loops aren't always bad if you can do a lot at once, and his code proves
> that you can have some fast loops.
>
> So what defines a slow loop? Is it having a bunch of accesses to
> sub-elements of arrays? Is it just having a bunch of statments? I suppose
> I could do some tests of my own, and I have a little, but it's much more
> fun to hear what you all have to say on the subject. I wouldn't have seen
> any IDL-ku if I just kept my thoughts to myself!
>
> Todd
Simple and short answer:
Loops are then evil, when you can avoid them.
cheers,
:-) marc

Subject: Re: Philosophy of for loops
Posted by landsman on Tue, 29 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Todd Clements wrote:
>>
>> Hello, again, everyone!
>>
>> I was just wondering what the general concensus of the "IDL Expert
>> Programmers" was on the use of for loops. When I first learned IDL,

A related question is when is it better to write loops as a single line,
and when is it better to use a DO BEGIN construct. For example,

inarr= randomn(seed, 3, 2048,2048)

outarr = fltarr(2048,2048,/nozero)

(1) for j=0,2047 do for i=0,2047 do outarr[i,j] = median(inarr[*,i,j])

(2) for j=0,2047 do begin
    for i=0,2047 do begin
        outarr[i,j] = median(inarr[*,i,j])
    endfor
    endfor

Form (1) is slightly faster, but the calculation cannot be interrupted
with a Control-C. Also, it is my impression that the speed difference
is less than it used to be, and that form (2) is now better optimized.

(I also assume that the two FOR loops are unavoidable here, but I would
be delighted to be proved wrong.)

--Wayne Landsman landsman@mpb.gsfc.nasa.gov

Sent via Deja.com http://www.deja.com/
Before you buy.

Subject: Re: Philosophy of for loops
Posted by davidf on Tue, 29 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

John-David Smith (jdsmith@astro.cornell.edu) writes:

> Despite what many supposedly learned men of
> the cloth will say to the contrary, for loops
> are without question the embodiment of pure evil.

Is it my imagination, or has the level of discourse
on this newsgroup gone up 5 or 6 notches in the past
couple of weeks?

Cheers,

David

P.S. Let's just say, I don't care *what* my
wife says, I'm *still* going to be checking the
newsgroup "about every five minutes". :-)

--
David Fanning, Ph.D.

Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Philosophy of for loops
Posted by Nando Iavarone on Tue, 29 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
Craig Markwardt wrote:
<blockquote TYPE=CITE>tclement@ucsd.edu (Todd Clements) writes:

>

> So what defines a slow loop? Is it having a bunch of accesses to

> sub-elements of arrays? Is it just having a bunch of statments? I
suppose

> I could do some tests of my own, and I have a little, but it's much
more

> fun to hear what you all have to say on the subject. I wouldn't have
seen

> any IDL-ku if I just kept my thoughts to myself!
<p>A good question. I think there are two parts to it, and you are
on

the right track. Theory comes first, then some practical solutions.
<p>IDL is a scripted language, so almost by definition very little

optimization can be done. Sure, the IDL code is "compiled," but
what

that really means is that the IDL statements are converted to some

intermediate form, sometimes called bytecode.</blockquote>
A Research Systems Engineer told me that it is called "P code".
<pre>--
Nando Iavarone
Advanced Computer System - SPACE DIVISION
via Lazzaro Belli, 23
00040 Frascati - RM
Tel: +39-6-944091 (switchboard)
            9440968 (direct)
E-mail:
    f.iavarone@acsys.it
    FrdndVrn@altavista.net</pre>
 </html>

Subject: Re: Philosophy of for loops
Posted by Martin Schultz on Tue, 29 Aug 2000 07:00:00 GMT

Todd Clements wrote:
>
> Hello, again, everyone!
>
> I was just wondering what the general concensus of the "IDL Expert
> Programmers" was on the use of for loops. When I first learned IDL, I
> remember getting from someone or somewhere the mantra "for loops are evil"
> because they take up so much time. Of course, as I learn more and watch
> what goes on in this group, it seems like "for loops are sometimes evil"
> would be a better mantra. The question then becomes, when do they become
> evil?
>
> In response to my thread on summing diagonal elements, Craig said that for
> loops aren't always bad if you can do a lot at once, and his code proves
> that you can have some fast loops.
>
> So what defines a slow loop? Is it having a bunch of accesses to
> sub-elements of arrays? Is it just having a bunch of statments? I suppose
> I could do some tests of my own, and I have a little, but it's much more
> fun to hear what you all have to say on the subject. I wouldn't have seen
> any IDL-ku if I just kept my thoughts to myself!
>
> Todd

I think it is mostly an economic problem. If it takes you 5
minutes to write code with a loop and that code takes 30 minutes
to execute, you can use these 30 minutes to buy and sell stock
and get rich (or poor ;-). If you use these 30 minutes to write
code without a loop which takes 5 minutes to run, you have spent
the same time without the opportunity. But then, if you want to
(or need to) run the code 100000L times, it will take a year if
the only loop is the outer one (for i=0L,99999 do ... - the L is
important!), but almost 6 years in the slow version. Just imagine
how many new releases of IDL you will miss if you run 6 years!

Cheers,
Martin

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie
[[
[[Bundesstr. 55, 20146 Hamburg
[[

[[phone: +49 40 41173-308
[[
[[fax: +49 40 41173-298
[[
[[martin.schultz@dkrz.de
[[
[[[[[[[[[[