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
 -- --------------