Subject: Re: A case where FOR loops are unavoidable?
Posted by JD Smith on Mon, 27 Nov 2006 22:44:36 GMT
View Forum Message <> Reply to Message

On Mon, 27 Nov 2006 14:19:16 -0800, gknoke wrote:

```
>  Hi all, I'm trying to setup an array for the computation of a rather
>  tricky multidimensional function, but I'm seeing no way around using
>  FOR loops in its creation.  I've been trying to figure out an
>  appropriate vector solution, but so far I'm drawing a blank.
>
>  The equation goes as follows:
>
>  f(x, b1, b2, p) = alog((1/b1)*(1-p)*exp(-x/b1) + (1/b2)*p*exp(-x/b2))
>
>  Here x is a vector roughly of length 2^20, b1 and b2 are roughly
>  100-1000 elements, and p is roughly 10-50 elements.  I realize that
>  this is a larger array than is likely to fit in memory, but what I'm
>  really after is the sum of this function in the x dimension, i.e. the
>  final output should be something like:
>
>  f_out(b1,b2,p) = total(f(x,b1,b2,p), 1)
>
>  My approach to this is to loop over b1, b2, and p to take advantage of
>  being able to use the total function to keep the array size manageable,
>  like so:
>
>  for k=1,np
>     for i=1,nb1
>        for j=1,nb2
>           f_out(i,j,k) = total(f(x, b1[i], b2[j], p[k]), 1)
>  etc. etc.
>
>  This works, but I'm wondering if there are any better approaches?
```

You could vectorize the whole thing by making an nx x nb1 x nb2 x np array and totaling in the correct dimension, but a) it won't fit in memory so it will page to disk, and b) this looping method is probably already about as fast as it gets (other than redesigning your algorithm). If the contents of the inner loop take much longer to execute than the looping penalty, removing loops doesn't help much if at all, and in many cases as mentioned can get you into trouble with memory. The looping penalty isn't huge, typically around .1 to .5 microseconds, but that can be very noticeable where inner loop contents also take on the order of a microsecond. It's 10-100x longer than the overhead of a tight loop in C.

JD

Subject: Re: A case where FOR loops are unavoidable?
Posted by news.qwest.net on Mon, 27 Nov 2006 23:26:50 GMT
View Forum Message <> Reply to Message

"gknoke" <greg.knoke@gmail.com> wrote in message
news:1164665956.774256.277610@14g2000cws.googlegroups.com...
> f(x, b1, b2, p) = alog((1/b1)*(1-p)*exp(-x/b1) + (1/b2)*p*exp(-x/b2))
>
> Here x is a vector roughly of length 2^20, b1 and b2 are roughly
> 100-1000 elements, and p is roughly 10-50 elements.

I don't understand what you mean with:
x/b1
where x has a million points, and b1 has 100 points.
I guess you mean that you are creating a 4 dimensional surface,
that has dimensions of [2^20,1000,1000,50].
That would seem to require 50 terrabytes of numbers, so 200 terrabytes
of data storage or so. Do you really need all those numbers?
How are you storing the results?

Perhaps using some of the well known algorithms to search parameter
space would be more appropriate than calculating the entire thing.

Anyways, to answer your question, vectorize on the largest vectors, and
loop over the smallest vectors, (which is what you are doing by sending
'x' in as a vector). If you can fit it in memory, you can do a two
dimensional
slice by also vectorizing b1, b2, or p (which fits in memory best).
for instance (this removes 2 loops)

```
for k=1,np
    for j=1,nb2
       f_out(i,j,k) = total(f(x, b1, b2[j], p[k]), 1)
```

If you really need the speed, perhaps switching over to fortran would help,
since this is a very simple calculation.

Cheers,
bob