Subject: Re: Was: Index... Now: Vectorize, huh?
Posted by Craig Markwardt on Thu, 05 Apr 2001 20:13:46 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" <pavel.romashkin@noaa.gov> writes:

> Craig and Med,
>
> I appreciate it! This is exactly what I needed. My problem is the lack
> of matrix operations knowledge. Craig's generic solution is exactly what
> I expected to see from Craig :-)

You're welcome.

```
> By the way. We all are big on vectorizing things in IDL. But look at this:
>
> IDL> a = test(2000)
>        1.4968959
> IDL> a = test(2000, /v)
>        3.2976190
>
```

> where TEST is below. I don't even mention that /VEC causes extremely
> high memory usage and gets totally out of hand on my system if S > 5000
> or so.

Yes, you can go too far overboard with vectorization. Part of the
problem is that you coded your vectorized path inefficiently. However
I think that when your matrices start to get huge, then the benefits
of vectorization can actually degrade, especially when you need to
artificially promote vectors into matrices as you are doing.

I've always said that if you can vectorize the inner loop of your
operation then you are usually fine. Pavel, you actually did that in
your *non*-vectorized case. :-)

A new version of TEST improves things slightly, but doesn't tip the
scales. w/ your version I get 1.6 and 3.8 s. With my version I get
1.5 and 2.7 s.

Craig

```
pro test, s, vec=vec
start = systime(1)
x = findgen(s)
if keyword_set(vec) then begin
  a = rebin(x,s,s)^2
  a = sqrt(transpose(a) + a)
endif else begin
```

```
  a = fltarr(s, s)
  for i = 0, s-1 do begin
    a[0, i] = sqrt(x^2 + i^2.)
  endfor
endelse
print, systime(1) - start
end
```

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

Subject: Re: Was: Index... Now: Vectorize, huh?
Posted by Stein Vidar Hagfors H[1] on Thu, 05 Apr 2001 21:11:38 GMT

View Forum Message <> Reply to Message

"Pavel A. Romashkin" <pavel.romashkin@noaa.gov> writes:

```
>  Craig and Med,
>
>  I appreciate it! This is exactly what I needed. My problem is the lack
>  of matrix operations knowledge. Craig's generic solution is exactly what
>  I expected to see from Craig :-)
>
>  By the way. We all are big on vectorizing things in IDL. But look at this:
>
>  IDL> a = test(2000)
>        1.4968959
>  IDL> a = test(2000, /v)
>        3.2976190
>
>  where TEST is below. I don't even mention that /VEC causes extremely
>  high memory usage and gets totally out of hand on my system if S > 5000
>  or so.
>
>  ;******************************
>  pro test, s, vec=vec
>  start = systime(1)
>  x = findgen(s)
>  a = fltarr(s, s)
>  if keyword_set(vec) then begin
>  a = sqrt(transpose(rebin(x, s, s))^2 + rebin(x, s, s)^2)
```

Now, this is a bit unfair! First (but least), you're generating a useless "a =
fltarr.." for the vector case. Then, you're doing everything in the wrong
order (exploding the array first, then squaring it), and you're also using
a transpose when it is not necessary:

```
  x2 = x^2 ; This could be used to speed up your "nonvectorized" code as well
  a = sqrt(rebin(reform(x2,1,s,/overwrite), s, s) + $
         rebin(reform(x2,s,1,/overwrite), s, s))
```

(But still, non-vectorizing code is 2x faster (with the x2 = x^2
optimization): Running through the large arrays several times is the killer, I
suspect. It may be architecture dependant, I presume).

But then, your "nonvectorized" code *is* vectorized (x is a vector), and it's
generally accepted that avoiding loops don't really buy you much when
operating on very large units. It's an intriguing example, though!

And for e.g. s = 200 the vectorized code is slightly faster..

```
> endif else begin
> for i = 0, s-1 do begin
> a[0, i] = sqrt(x^2 + i^2.)
> endfor
> endelse
> print, systime(1) - start
> ;return, a
> end
> ;*****************************
```

--
Stein Vidar Hagfors Haugan
ESA SOHO SOC/European Space Agency Science Operations Coordinator for SOHO

Subject: Re: Was: Index... Now: Vectorize, huh?
Posted by Pavel A. Romashkin on Thu, 05 Apr 2001 21:13:06 GMT

View Forum Message <> Reply to Message

Craig Markwardt wrote:

> Part of the
> problem is that you coded your vectorized path inefficiently.

I tried it the way you did, but decided to try the full "one-line"
vectorized solution.

> I've always said that if you can vectorize the inner loop of your
> operation then you are usually fine. Pavel, you actually did that in

> your *non*-vectorized case. :-)

This is true, and I do this in my code since I found that out a while
ago. I wonder if that means, in fact, that array operations in IDL are
optimized for vectors (row-wise arrays) only, and once you are into more
than 1 dimension, you are better off looping through other dimensions.

> A new version of TEST improves things slightly, but doesn't tip the
> scales. w/ your version I get 1.6 and 3.8 s. With my version I get
> 1.5 and 2.7 s.

On my system, your own solution from the previous post is faster still,
but the loop can not be defeated:

```
;***********
function test, s, vec=vec
start = systime(1)
x = findgen(s)
if keyword_set(vec) then begin
;a = rebin(x, s, s)^2
;a = sqrt(transpose(a) + a)
;a = sqrt((transpose(rebin(x, s, s)))^2 + (rebin(x, s, s))^2)
a = (x # (fltarr(s)+1))^2
a = sqrt(transpose(a) + a)
endif else begin
a = fltarr(s, s)
for i = 0, s-1 do a[0, i] = sqrt(x^2 + i^2.)
endelse
print, systime(1) - start
return, a
end
;***********
```

Cheers,
Pavel

Subject: Re: Was: Index... Now: Vectorize, huh?
Posted by Pavel A. Romashkin on Thu, 05 Apr 2001 21:49:15 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:

> Now, this is a bit unfair! First (but least), you're generating a useless "a =
> fltarr.." for the vector case. Then, you're doing everything in the wrong
> order (exploding the array first, then squaring it), and you're also using
> a transpose when it is not necessary:

Well, if nothing else, this thread can be used as an example of how not

to write programs! I admit, I did not think much when I wrote that
example function, partially because I was really surprised by the
large time gap in the results. Not saying that if I did think that
would've changed anything :-(
Oh well. At least we've justified the loops completely now. And pulled
Stein Vidar ouf of hiding, too :-)

Cheers,
Pavel