Subject: vectorization challenge! (help!)
Posted by Conor on Tue, 17 Jul 2007 16:52:24 GMT
View Forum Message <> Reply to Message

I'm 'vectorizing' a piece of code to speed it up. It's part of a larger program. One of the sections is turning out to be very difficult to vectorize (am I using that word right? What I mean is I'm trying to get rid of for loops). Anyway, maybe someone has some thoughts on how to vectorize it. Maybe it's just not worth it for this section. Here's the basic idea, filled in with dummy data:

```
n = 24
npeeps = 50
gn1 = findgen(n,npeeps)
gn2 = findgen(n,npeeps)
cutoff = .01

for i=0,npeeps-1 do begin

    ; make a random value to determine if we do anything with this
row
    if randomu(seed,1) lt cutoff then begin

        ; this row has been selected. Swap the last (random number)
of digits in gn1[*,i] with gn2[*,i]
        randindex = long(randomu(seed,1)*n*nd)
        temp = gn1[randindex:*,i]
        gn1[randindex:*,i] = gn2[randindex:*,i]
        gn2[randindex:*,i] = temp

    endif

endfor
```

That's it. For randomly selected rows, swap a random number of elements at the end of the row with another array. It is surpisingly difficult to get rid of that for loop. Maybe I'm just a bit out of it today though. I thought of generating a list of indexes to be swapped, but I can't quite figure it out. Oh, if only IDL allowed the syntax: arr[st:ed] where st and ed are arrays themselves! Then this would be really easy (something like this would do it):

```
st = long(randomu(seed,npeeps)*n*nd)
ed = make_array(npeeps,/integer,value=n)
indlist = indgen(n)
inds = indlist[st:ed] + indgen(npeeps)*n
temp = gn1[inds]
gn1[inds] = gn2[inds]
```

gn2[inds] = temp

Alas, indlist[st:ed] isn't allowed! (Also, indgen(npeeps)*n has the
wrong dimensions anyway...)

Subject: Re: vectorization challenge! (help!)
Posted by Brian Larsen on Wed, 18 Jul 2007 20:37:40 GMT
View Forum Message <> Reply to Message

There is at least some help I can offer here.

instead of:
      if randomu(seed,1) lt cutoff then begin
you can make an array of the random numbers to use in the decision as
a mask and take it out of the loop
    swap_mask = randomu(seed, npeeps) lt cutoff
this has the advantage f being a byte array so it is small.

Once you have this mask you only need to loop over the 1's in the mask
which is at least fewer steps.

That is as much as my brain has this second but maybe this will spark
something for you.

Brian

---------- ----------
Brian Larsen
Boston University
Center for Space Physics

Subject: Re: vectorization challenge! (help!)
Posted by alvin[1] on Thu, 19 Jul 2007 08:01:55 GMT
View Forum Message <> Reply to Message

On Jul 17, 9:52 pm, Conor <cmanc...@gmail.com> wrote:
> I'm 'vectorizing' a piece of code to speed it up. It's part of a
> larger program. One of the sections is turning out to be very
> difficult to vectorize (am I using that word right? What I mean is
> I'm trying to get rid of for loops). Anyway, maybe someone has some
> thoughts on how to vectorize it. Maybe it's just not worth it for
> this section. Here's the basic idea, filled in with dummy data:
>

```
>  n = 24
>  npeeps = 50
>  gn1 = findgen(n,npeeps)
>  gn2 = findgen(n,npeeps)
>  cutoff = .01
>
>  for i=0,npeeps-1 do begin
>
>      ; make a random value to determine if we do anything with this
>  row
>      if randomu(seed,1) lt cutoff then begin
>
>          ; this row has been selected.  Swap the last (random number)
>  of digits in gn1[*,i] with gn2[*,i]
>          randindex = long(randomu(seed,1)*n*nd)
>          temp = gn1[randindex:*,i]
>          gn1[randindex:*,i] = gn2[randindex:*,i]
>          gn2[randindex:*,i] = temp
>
>      endif
>
>  endfor
>
>  That's it.  For randomly selected rows, swap a random number of
>  elements at the end of the row with another array.  It is surpisingly
>  difficult to get rid of that for loop.  Maybe I'm just a bit out of it
>  today though.  I thought of generating a list of indexes to be
>  swapped, but I can't quite figure it out.  Oh, if only IDL allowed the
>  syntax: arr[st:ed] where st and ed are arrays themselves!  Then this
>  would be really easy (something like this would do it):
>
>  st = long(randomu(seed,npeeps)*n*nd)
>  ed = make_array(npeeps,/integer,value=n)
>  indlist = indgen(n)
>  inds = indlist[st:ed] + indgen(npeeps)*n
>  temp = gn1[inds]
>  gn1[inds] = gn2[inds]
>  gn2[inds] = temp
>
>  Alas, indlist[st:ed] isn't allowed!  (Also, indgen(npeeps)*n has the
>  wrong dimensions anyway...)
```

Hi there:

If I understand your problem truly, I would have done this:
The code is not efficient, and contains redundant values in the 'for'
loop.

With a little playing around, you can reduce the size of randindex in the loop.

Alvin

```
n=24        ; How large is this value?
npeeps=50    ;How large is this?
gn1 = findgen(n,npeeps)
gn2=qn1

;;;;;cutoff = .01 ;;;; What does this do?

randindex = long(randomu(seed,npeeps)*n)          ;;;;; I don't know
what 'nd' is!

;;;if randomu(seed,1) lt cutoff then begin  ??    ;;;;This is throwing
me off...
                                          ;;;;What do you
intend to do here?

thisval=n*indgen(npeeps)+n-1
for i=1,n do randindex=[randindex,(randindex+n-1)<thisval]
temp = gn1[randindex]
gn1[randindex] = gn2[randindex]
gn2[randindex] = temp
```

Subject: Re: vectorization challenge! (help!)
Posted by Maarten[1] on Thu, 19 Jul 2007 08:35:50 GMT
View Forum Message <> Reply to Message

On Jul 17, 4:52 pm, Conor <cmanc...@gmail.com> wrote:

```
>  n = 24
>  npeeps = 50
>  gn1 = findgen(n,npeeps)
>  gn2 = findgen(n,npeeps)
>  cutoff = .01
>
>  for i=0,npeeps-1 do begin
>
>      ; make a random value to determine if we do anything with this
>  row
>      if randomu(seed,1) lt cutoff then begin
>
>          ; this row has been selected.  Swap the last (random number)
>  of digits in gn1[*,i] with gn2[*,i]
```

```
>           randindex = long(randomu(seed,1)*n*nd)
>           temp = gn1[randindex:*,i]
>           gn1[randindex:*,i] = gn2[randindex:*,i]
>           gn2[randindex:*,i] = temp
>
>       endif
>
>  endfor
```

I don't think you can get rid of the for loop, but you can get rid of
the if statement, and shorten the for loop by using

```
ran_levels = randu(seed,npeeps)
idx = where(ran_levels lt cutoff, cnt)
for i=0,cnt-1 do
   randindex = long(randomu(seed,1)*n*nd)
   temp = gn1[randindex:*,idx[i]]
   gn1[randindex:*,idx[i]] = gn2[randindex:*,idx[i]]
   gn2[randindex:*,idx[i]] = temp
endfor
```

> From here you may be able to use the tricks listed in
http://www.dfanning.com/code_tips/drizzling.html

Hope this helps.

Maarten

Subject: Re: vectorization challenge! (help!)
Posted by mattf on Thu, 19 Jul 2007 11:53:49 GMT
View Forum Message <> Reply to Message

On Jul 17, 12:52 pm, Conor <cmanc...@gmail.com> wrote:

```
>  I'm 'vectorizing' a piece of code to speed it up.  It's part of a
>  larger program.  One of the sections is turning out to be very
>  difficult to vectorize (am I using that word right?  What I mean is
>  I'm trying to get rid of for loops).  Anyway, maybe someone has some
>  thoughts on how to vectorize it.  Maybe it's just not worth it for
>  this section.  Here's the basic idea, filled in with dummy data:
>
>  n = 24
>  npeeps = 50
>  gn1 = findgen(n,npeeps)
>  gn2 = findgen(n,npeeps)
>  cutoff = .01
>
>  for i=0,npeeps-1 do begin
```

```
>
>      ; make a random value to determine if we do anything with this
>  row
>      if randomu(seed,1) lt cutoff then begin
>
>          ; this row has been selected.  Swap the last (random number)
>  of digits in gn1[*,i] with gn2[*,i]
>          randindex = long(randomu(seed,1)*n*nd)
>          temp = gn1[randindex:*,i]
>          gn1[randindex:*,i] = gn2[randindex:*,i]
>          gn2[randindex:*,i] = temp
>
>      endif
>
>  endfor
>
>  That's it.  For randomly selected rows, swap a random number of
>  elements at the end of the row with another array.  It is surpisingly
>  difficult to get rid of that for loop.  Maybe I'm just a bit out of it
>  today though.  I thought of generating a list of indexes to be
>  swapped, but I can't quite figure it out.  Oh, if only IDL allowed the
>  syntax: arr[st:ed] where st and ed are arrays themselves!  Then this
>  would be really easy (something like this would do it):
>
>  st = long(randomu(seed,npeeps)*n*nd)
>  ed = make_array(npeeps,/integer,value=n)
>  indlist = indgen(n)
>  inds = indlist[st:ed] + indgen(npeeps)*n
>  temp = gn1[inds]
>  gn1[inds] = gn2[inds]
>  gn2[inds] = temp
>
>  Alas, indlist[st:ed] isn't allowed!  (Also, indgen(npeeps)*n has the
>  wrong dimensions anyway...)
```

There are a number of vectorization strategies-- one is heavy use of the WHERE() function. Given a condition C for an action on elements some vector A, you can set A[WHERE(C)] to whatever you want.

Another trick is to treat a matrix as a (long) one dimensional vector and then make use of modular arithmetic. But be careful when you go back to matrix-land with the REFORM() function-- chances are you will get rows and columns backwards.

A third vectorization trick is to exchange a FOR loop for N copies of your variables all concatenated together, and then doing the N calculations with matrix arithmetic. This is a last-ditch vectorization strategy that depends on exchanging the processing time

in a FOR loop for a no-branching calculation that wastes a lot of
memory

Subject: Re: vectorization challenge! (help!)
Posted by Brian Larsen on Thu, 19 Jul 2007 13:33:35 GMT
View Forum Message <> Reply to Message

Odd, I tried to answer this yesterday and it got lost in the bits I
guess.

I have one suggestion that I often use that might kick start your
thinking on this.

instead of doing this in the loop
    if randomu(seed,1) lt cutoff then begin
try making a mask outside of the loop
    mask = randomu(seed, npeeps) lt cutoff
this returns a byte array (so its small) of 1's and 0's then at worst
you only have to loop over the 1's or at best can use where to just
perform an operation on the 1's.

I find that this kind of thing is often the secret to fixing this kind
of code.

Cheers,

Brian

---------- ----------
Brian Larsen
Boston University
Center for Space Physics

Subject: Re: vectorization challenge! (help!)
Posted by alvin[1] on Thu, 19 Jul 2007 16:40:09 GMT
View Forum Message <> Reply to Message

Hi there:

If I understand your problem truly, I would have done this:
The code is not efficient, and contains redundant values in the 'for'
loop.
With a little playing around, you can reduce the size of randindex in
the
loop. Hope this helps.

Alvin Das

```
n=24        ; How large is this value?
npeeps=50    ;How large is this?
gn1 = findgen(n,npeeps)
gn2=qn1

;;;;;cutoff = .01 ;;;; What does this do?

randindex = long(randomu(seed,npeeps)*n)          ;;;;; I don't know
what 'nd'
is!

;;;if randomu(seed,1) lt cutoff then begin  ??    ;;;;This is throwing
me
off...
                                          ;;;;What do you
intend to do
here?

thisval=n*indgen(npeeps)+n-1
for i=1,n do randindex=[randindex,(randindex+n-1)<thisval]
temp = gn1[randindex]
gn1[randindex] = gn2[randindex]
gn2[randindex] = temp
```

Subject: Re: vectorization challenge! (help!)
Posted by MarioIncandenza on Thu, 19 Jul 2007 17:56:51 GMT
View Forum Message <> Reply to Message

Let's throw some memory at the problem, shall we?
Note: this is completely untested.

```
n = 24
npeeps = 50
gn1 = findgen(n,npeeps)
gn2 = findgen(n,npeeps)
cutoff = .01
; here is the array-based version of your task
swap=randomu(seed,n,npeeps)
bswap=swap le cutoff; binary
nswap_per_row=total(bswap,1); number to swap in each row
nswap_mask=rebin(transpose(nswap_per_row),n,npeeps); each cell has N
to swap in that row

swapi=reverse(lindgen(n,npeeps) mod n),1); each row N-1 to 0
```

bswapi=swapi lt nswap_mask; if NSWAP_PER_ROW=N, swaps entire row

fswap=where(bswapi,nfswap)
if(nswap gt 0) then gn1[fswap] = gn2[fswap]