Subject: Re: simple vectorizing problem
Posted by dpm314 on Tue, 09 Sep 2008 01:48:42 GMT
View Forum Message <> Reply to Message

On Sep 8, 8:47 pm, dpm314 <meich...@gmail.com> wrote:
> Hey - here is a vectorizing question from an experienced programmer
> 'newish' to IDL.
>
> I have a stack of images dimensions (nx,ny, num_images) , a LOT of
> data sometimes, on which I need to preform a whole mathematical
> operation on. For example, one thing I need to do is create an array
> of the mean of each image.
>
> I've done this successfully with something like
> averages = fltarr(num_images)
> for k = 0, NUM_OF_CLICKS-1 do averages[k] = mean(image(*,*,k)
>
> which makes perfect sense to a c/c++/Fortran programmer. But, this
> really hangs up for the size images I am working with (512*512*70000
> or more). Is there a way to vectorize? I've tried several things
> like:
>
> averages = fltarr(num_images)
> a = indgen(num_images)
> averages(a) = mean(image(*,*,a))
>
> I've also tried reforming the image to by 2D, with dimensions [nx*ny,
> num_images] and doing a similar thing, but no luck. That included
> making an array of integers, call it b, where b = 0, nx*ny*1 - 1,
> nx*ny*2 - 1, nx*ny*3 - 1 ... and doing something like
> averages(a) = mean(image(b[a]:b[a+1], a)
> (I know this probably will run out of bounds, I can't remember right
> now what I did, but it still didn't work and I had the series worked
> out right...)
>
> It seems that in a language like IDL there should be a way to do
> something like this without writing a for-loop.
>
> Thanks,
>
> David M.

Sorry, in the first bit of code NUM_OF_CLICKS is same thing as
num_images ... David M.

Subject: Re: simple vectorizing problem

dpm314 writes:

> Hey - here is a vectorizing question from an experienced programmer
> 'newish' to IDL.
>
> I have a stack of images dimensions (nx,ny, num_images) , a LOT of
> data sometimes, on which I need to preform a whole mathematical
> operation on. For example, one thing I need to do is create an array
> of the mean of each image.
>
> I've done this successfully with something like
> averages = fltarr(num_images)
> for k = 0, NUM_OF_CLICKS-1 do averages[k] = mean(image(*,*,k)
>
> which makes perfect sense to a c/c++/Fortran programmer. But, this
> really hangs up for the size images I am working with (512*512*70000
> or more). Is there a way to vectorize? I've tried several things
> like:
>
> averages = fltarr(num_images)
> a = indgen(num_images)
> averages(a) = mean(image(*,*,a))
>
> I've also tried reforming the image to by 2D, with dimensions [nx*ny,
> num_images] and doing a similar thing, but no luck. That included
> making an array of integers, call it b, where b = 0, nx*ny*1 - 1,
> nx*ny*2 - 1, nx*ny*3 - 1 ... and doing something like
> averages(a) = mean(image(b[a]:b[a+1], a)
> (I know this probably will run out of bounds, I can't remember right
> now what I did, but it still didn't work and I had the series worked
> out right...)
>
> It seems that in a language like IDL there should be a way to do
> something like this without writing a for-loop.

Alright, I'll take a stab at this. There is no tennis or election
coverage on the TV anyway.

```
theImageMeans = Total(Reform(images, nx*ny, num_images), 1) / (nx*ny)
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: simple vectorizing problem
Posted by Chris[6] on Tue, 09 Sep 2008 08:50:07 GMT
View Forum Message <> Reply to Message

Looks like the memory allocation overhead associated with
vectorization is worse than looping:

vectorization 1 time: 5.0360808 s
vectorization 2 time: 4.8453941 s
looping itme: 4.0875931 s

The vectorization performances become comparitvely worse as the array
grows (both reform() and total() create temporary, large arrays).

chris

```
pro test

nimage = 1500
im = fltarr(512,512,nimage)

t0 = systime(/seconds)
mean=total(reform(im,512L*512L, nimage), 1)/(512.*512)
t1=systime(/seconds)

mean=total(total(im,1),1)/(512.*512.)

t2 = systime(/seconds)

for i=0, nimage-1, 1 do $
  mean[i]=total(im[*,*,i])/(512.*512)

t3=systime(/seconds)

print, t1-t0
print, t2-t1
print, t3-t2

end
```

Subject: Re: simple vectorizing problem
Posted by Foldy Lajos on Tue, 09 Sep 2008 09:58:15 GMT

On Tue, 9 Sep 2008, Chris wrote:

> Looks like the memory allocation overhead associated with
> vectorization is worse than looping:
>
> vectorization 1 time: 5.0360808 s
> vectorization 2 time: 4.8453941 s
> looping itme: 4.0875931 s
>
> The vectorization performances become comparitvely worse as the array
> grows (both reform() and total() create temporary, large arrays).
>
> chris
>
> pro test
>
> nimage = 1500
> im = fltarr(512,512,nimage)
>
> t0 = systime(/seconds)
> mean=total(reform(im,512L*512L, nimage), 1)/(512.*512)
> t1=systime(/seconds)
>
> mean=total(total(im,1),1)/(512.*512.)
>
> t2 = systime(/seconds)
>
> for i=0, nimage-1, 1 do $
> mean[i]=total(im[*,*,i])/(512.*512)
>
> t3=systime(/seconds)
>
> print, t1-t0
> print, t2-t1
> print, t3-t2
>
> end
>
>

try this:

```
t0 = systime(/seconds)
mean=total(reform(im,512L*512L, nimage, /overwrite), 1)/(512.*512)
;                               ^^^^^^^^^^^^
t1=systime(/seconds)
```

regards,
lajos

Subject: Re: simple vectorizing problem
Posted by dpm314 on Wed, 10 Sep 2008 23:53:43 GMT
View Forum Message <> Reply to Message

On Sep 9, 4:58 am, FÖLDY Lajos <fo...@rmki.kfki.hu> wrote:
> On Tue, 9 Sep 2008, Chris wrote:
>> Looks like the memory allocation overhead associated with
>> vectorization is worse than looping:
>
>> vectorization 1 time: 5.0360808 s
>> vectorization 2 time: 4.8453941 s
>> looping itme: 4.0875931 s
>
>> The vectorization performances become comparitvely worse as the array
>> grows (both reform() and total() create temporary, large arrays).
>
>> chris
>
>> pro test
>
>> nimage = 1500
>> im = fltarr(512,512,nimage)
>
>> t0 = systime(/seconds)
>> mean=total(reform(im,512L*512L, nimage), 1)/(512.*512)
>> t1=systime(/seconds)
>
>> mean=total(total(im,1),1)/(512.*512.)
>
>> t2 = systime(/seconds)
>
>> for i=0, nimage-1, 1 do $
>> mean[i]=total(im[*,*,i])/(512.*512)
>
>> t3=systime(/seconds)
>
>> print, t1-t0
>> print, t2-t1
>> print, t3-t2
>
>> end
>

> try this:
>
> t0 = systime(/seconds)
> mean=total(reform(im,512L*512L, nimage, /overwrite), 1)/(512.*512)
> ; ^^^^^^^^^^^^
> t1=systime(/seconds)
>
> regards,
> lajos

Hi everyone and thanks a lot for your suggestions so far. That is a cute trick with the Total function, and I was just not aware that you could specify to total only rows or columns like that. This will work, and I have tried it but didn't get a chance to do the timing test.
However, my first post must not be clear, because this doesn't really answer my question.

I have these image arrays and need to perform several functions on them which return a scalar (or a 6 element array) for each image in a large stack. Is there *in general* a way to so something like:

results = fltarr(num_images)
results = myFunctThatDoesSomeAnalysis(image)
;now results holds an array of scalars for the result of myFunctThatDoesSomeAnalysis() on each image

without either using a loop here or inside the analysis function iterating over each frame in the image?

again, I've tried many things like:

a = indgen(num_image)
results(a) = myFunctThatDoesSomeAnalysis(image(*,*,a))
which I thought would work but it does not.

It is necessary to use a loop like this then in general :
for i 0, num_images-1 do results(i) =
myFunctThatDoesSomeAnalysis(image(*,*,i))

???

I thought the trick with an indexing array ('a' in the example above) would work because if I say
b = indgen(101)/(!pi/100)
print, cos(b)

I get not one number out but an array of cos() evaluated from 0 to pi

I guess I don't understand how the cos() example gives me an array
back, and
 myFunctThatDoesSomeAnalysis(image(*,*,a)) gives me just one number.

If this question makes no sense I apologize, and I could provide more
code to illustrate the problem if that would help. I am still trying
to wrap my head around vectorizing code, which (I at least) find quite
different from purely procedural languages like c and Fortran which
I've written in for years.

Thanks,
David M.

Subject: Re: simple vectorizing problem
Posted by David Fanning on Thu, 11 Sep 2008 00:14:17 GMT
View Forum Message <> Reply to Message

dpm314 writes:

> I have these image arrays and need to perform several functions on
> them which return a scalar (or a 6 element array) for each image in a
> large stack. Is there *in general* a way to so something like:
>
> results = fltarr(num_images)
> results = myFunctThatDoesSomeAnalysis(image)
> ;now results holds an array of scalars for the result of
> myFunctThatDoesSomeAnalysis() on each image
>
> without either using a loop here or inside the analysis function
> iterating over each frame in the image?

I still don't have a good idea about what you are asking for,
but I suspect now it is because I have never worked with
a modern language. This must be something you can do in
C##.

But, I do have a great deal of confidence in saying that
in general (depending, of course, on what exactly
myFunctThatDoesSomeAnalysis actually *does*) that you
are going to need a loop *somewhere*.

IDL is *not* a modern language. It's near 25 years old.
And while it has a few tricks up its sleeve, it still has
more similarities to BASIC and FORTRAN than to C++.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: simple vectorizing problem
Posted by Spon on Thu, 11 Sep 2008 12:15:11 GMT
View Forum Message <> Reply to Message

> I thought the trick with an indexing array ('a' in the example above)
> would work because if I say
> b = indgen(101)/(!pi/100)
> print, cos(b)
>
> I get not one number out but an array of cos() evaluated from 0 to pi
>
> I guess I don't understand how the cos() example gives me an array
> back, and
> myFunctThatDoesSomeAnalysis(image(*,*,a)) gives me just one number.

Simply put, the IDL function COS() has been specifically written so it
can take either scalars or arrays as arguments. If you want
myFunctThatDoesSomeAnalysis() to do the same, you have to write it in
such a way that it can also handle arrays.
Whether this can be done without expensive FOR loops depends entirely
on what your function does. If it can be done, it's probably
worthwhile doing it though.

> Thanks,
> David M.

Regards,
Chris

Subject: Re: simple vectorizing problem
Posted by JD Smith on Thu, 11 Sep 2008 18:56:50 GMT
View Forum Message <> Reply to Message

On Sep 10, 7:53 pm, dpm314 <meich...@gmail.com> wrote:
> Hi everyone and thanks a lot for your suggestions so far. That is a
> cute trick with the Total function, and I was just not aware that you
> could specify to total only rows or columns like that. This will

> work, and I have tried it but didn't get a chance to do the timing
> test.
> However, my first post must not be clear, because this doesn't really
> answer my question.
>
> I have these image arrays and need to perform several functions on
> them which return a scalar (or a 6 element array) for each image in a
> large stack. Is there *in general* a way to so something like:
>
> results = fltarr(num_images)
> results = myFunctThatDoesSomeAnalysis(image)
> ;now results holds an array of scalars for the result of
> myFunctThatDoesSomeAnalysis() on each image
>
> without either using a loop here or inside the analysis function
> iterating over each frame in the image?
>
> again, I've tried many things like:
>
> a = indgen(num_image)
> results(a) = myFunctThatDoesSomeAnalysis(image(*,*,a))
> which I thought would work but it does not.
>
> It is necessary to use a loop like this then in general :
> for i 0, num_images-1 do results(i) =
> myFunctThatDoesSomeAnalysis(image(*,*,i))
>
> ???
>
> I thought the trick with an indexing array ('a' in the example above)
> would work because if I say
> b = indgen(101)/(!pi/100)
> print, cos(b)
>
> I get not one number out but an array of cos() evaluated from 0 to pi
>
> I guess I don't understand how the cos() example gives me an array
> back, and
> myFunctThatDoesSomeAnalysis(image(*,*,a)) gives me just one number.
>
> If this question makes no sense I apologize, and I could provide more
> code to illustrate the problem if that would help. I am still trying
> to wrap my head around vectorizing code, which (I at least) find quite
> different from purely procedural languages like c and Fortran which
> I've written in for years.

There is no general mechanism to "thread" arbitrary code over
arbitrary dimensions within an array in IDL. Some languages have such

general operator threading (but they're not exactly fast). There *are* a variety of internal IDL routines which can do such threading, including TOTAL, and various others, which can be used together when vectorizing many types of problems (certainly more than are immediately obvious). Another way of saying that is that all IDL vectorization, aside from basic array arithmetic, is explicit, in the sense that it is specifically requested in the called procedure or function (like TOTAL(array, dimension)).

There is such a thing as premature vectorization, however. In your case, if the images in your stack are big enough (at least few hundred or thousand pixels), simply looping through each one and accumulating the result will likely perform just as well as any vectorized method could. It's when you're looping over individual array elements that you should be concerned.... "Embodiment of Pure Evil" notwithstanding.
