Subject: Array Concatenation Optimization
Posted by KM on Tue, 08 Feb 2005 19:19:32 GMT
View Forum Message <> Reply to Message

I've read JD's tutorial
[http://www.dfanning.com/tips/array_concatenation.html] a few times,
but cannot get my array juggling as fast as I would like it...

I'm converting an 8 bit image to 24 bit RGB. Why? Well, long story,
but I am stuck in the Z buffer and need to make nice anti-aliased
color images. So everything is 3x as large&thick, converted to RGB,
and then rebined...

The conversion to RGB is currently one of the bottlenecks in my
code, and I would like to speed it up. I am using this function:

```
function toRGB, r,g,b, image
s = size(image,/dim)
image_rgb = bytarr( 3, s[0], s[1] )
image_rgb[0, *, *] = r[image]
image_rgb[1, *, *] = g[image]
image_rgb[2, *, *] = b[image]
return, image_rgb
end
```

I can cut the execution time in half if I change the entire function
to this one line:

```
return, [[[r[image]]],[[g[image]]],[[b[image]]]]
```

But now it is [n,m,3], and the WRITE_PNG procedure needs it to be
[3,n,m], and wrapping a TRANSPOSE() around that 1 line makes it go
from 2x as fast to 7x as slow as the original function.

I don't get any improvement if I change the * to explicit ranges,
although I remember reading that this should make array insertions
faster....

```
image_rgb[0, 0:s[0]-1, 0:s[1]-1] = r[image]
```

Another trick I have read about is to use the TEMPORARY() function,
but I don't think it is applicable in this case.

Any other suggestions?

Thanks,

Ken Mankoff

http://edgcm.org/
http://spacebit.dyndns.org/

Subject: Re: Array concatenation
Posted by David Fanning on Wed, 07 Jun 2006 13:41:44 GMT
View Forum Message <> Reply to Message

maye writes:

> I was searching for an effective way to do this, but can't find anybody
> writing about what (as usual) seems to be an obvious problem/task to
> me. :)
> I don't know how many elements I will collect while scanning a bunch of
> images, so I want to use array concatenation to collect the values like
> this:
> means = [means, currMean]
> But for this to compile/run properly, I need mean to exist before.
> If I do a
> means = 0.
> before the loop, I will always have a zero-element in the beginning
> that I don't want at plotting time. Of course I could remove it with
> means = means[1:*]
> but not only does this waste resources, it also becomes cumbersome if I
> collect 20 different data values, for that I ALL have to remove the
> first value?
> Surely there must be a better way?
> Please help me to program IDL efficiently! :)

I think most people do something like this:

```
IF N_Elements(means) EQ 0 THEN means = [currMean] ELSE $
   means = [Temporary(means), currMean]
```

Or, did you mean something that doesn't involve any coding? :-)

Be aware that if your loop is large, it is much more
efficient to allocate the memory for your array in chucks
of say 100 or 1000, and then fill it up, then it is to
continuously re-define your array like this.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Array concatenation
Posted by Paul Van Delst[1] on Wed, 07 Jun 2006 14:21:47 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> maye writes:
>
>
>> I was searching for an effective way to do this, but can't find anybody
>> writing about what (as usual) seems to be an obvious problem/task to
>> me. :)
>> I don't know how many elements I will collect while scanning a bunch of
>> images, so I want to use array concatenation to collect the values like
>> this:
>> means = [means, currMean]
>> But for this to compile/run properly, I need mean to exist before.
>> If I do a
>> means = 0.
>> before the loop, I will always have a zero-element in the beginning
>> that I don't want at plotting time. Of course I could remove it with
>> means = means[1:*]
>> but not only does this waste resources, it also becomes cumbersome if I
>> collect 20 different data values, for that I ALL have to remove the
>> first value?
>> Surely there must be a better way?
>> Please help me to program IDL efficiently! :)
>
>
> I think most people do something like this:
>
> IF N_Elements(means) EQ 0 THEN means = [currMean] ELSE $
> means = [Temporary(means), currMean]
>
> Or, did you mean something that doesn't involve any coding? :-)
>
> Be aware that if your loop is large, it is much more
> efficient to allocate the memory for your array in chucks
> of say 100 or 1000, and then fill it up, then it is to
> continuously re-define your array like this.

Let me second David's statement. For many points, array concatenation is s..l..o..w. In my early IDL days I improved the run time of a procedure (to read a 100,000's->millions of points) from minutes[*] to effectively instantaneous by switching from concatenation to allocate a block and then reallocate (or trim) extra space as required. Now it takes longer to output the number of points read rather than reading them.

For a small number of points array concatenation is great, but it does not scale well.

paulv

[*] When I had to read many, many of the same type of files I could walk down the street
to get a coffee before it had finished.

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Array concatenation
Posted by David Fanning on Wed, 07 Jun 2006 14:33:44 GMT
View Forum Message <> Reply to Message

Paul Van Delst writes:

> [*] When I had to read many, many of the same type of files I could walk down the street
> to get a coffee before it had finished.

I'm beginning to think this could be an advantage
in a go-go world. :-)

Cheers,

David

P.S. The power went out at my house last night, and the
whole neighborhood was walking around with flashlights,
talking to each other. Even the kids had to talk to us,
until they stumbled onto a laptop with a DVD player. :-(

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Array concatenation
Posted by Michael Galloy on Wed, 07 Jun 2006 15:06:45 GMT
View Forum Message <> Reply to Message

maye wrote:
> Hi folks,
> I was searching for an effective way to do this, but can't find anybody
> writing about what (as usual) seems to be an obvious problem/task to
> me. :)

> I don't know how many elements I will collect while scanning a bunch of
> images, so I want to use array concatenation to collect the values like
> this:
> means = [means, currMean]
> But for this to compile/run properly, I need mean to exist before.
> If I do a
> means = 0.
> before the loop, I will always have a zero-element in the beginning
> that I don't want at plotting time. Of course I could remove it with
> means = means[1:*]
> but not only does this waste resources, it also becomes cumbersome if I
> collect 20 different data values, for that I ALL have to remove the
> first value?
> Surely there must be a better way?
> Please help me to program IDL efficiently! :)
> Best regards,
> Michael
>

I have a class MGArrayList that I use for such things. It follows the same interface as IDL_Container, but allows for any IDL data type (one data type per object). So for example,

```
IDL> olist = obj_new('mgarraylist', example=0.0, blocksize=10)
IDL> olist->add, 3.0
IDL> olist->add, 5.0
IDL> olist->add, 10.0
IDL> print, olist->get(/all)
      3.00000      5.00000      10.0000
```

There's more info along with links to download and API documentation at:

http://michaelgalloy.com/?p=11

Mike
--
www.michaelgalloy.com

Subject: Re: Array concatenation
Posted by FL on Wed, 07 Jun 2006 16:12:56 GMT

View Forum Message <> Reply to Message

Hi,

concatenation is much faster in FL. I can't help showing a comparison.
Take this simple test program:

```
n=10l
for i=1,8 do begin
   a=[0l]
   t=systime(1)
   for j=1l,n do a=[a,j]
   print, n, systime(1)-t
   n *= 10l
endfor
end
```

On an Opteron 142 linux system FL prints:

```
         10   1.5974045e-05
        100   6.7949295e-05
       1000   0.00091814995
      10000     0.051065922
     100000      0.12665701
    1000000       1.2369611
   10000000       12.450622
  100000000       123.65290
```

In IDL 6.2 I have set i=1,5 (for obvious reasons :-), and the output is:

```
         10   1.4066696e-05
        100   7.3909760e-05
       1000   0.00089907646
      10000     0.045171022
     100000       22.684564
```

(FL has its own memory allocator and allocation strategy, this is the
reason of the big difference.)

regards,
lajos

On Wed, 7 Jun 2006, Paul Van Delst wrote:

> Let me second David's statement. For many points, array concatenation is s..l..o..w. In my
> early IDL days I improved the run time of a procedure (to read a 100,000's->millions of
> points) from minutes[*] to effectively instantaneous by switching from concatenation to
> allocate a block and then reallocate (or trim) extra space as required. Now it takes
> longer to output the number of points read rather than reading them.
>
> For a small number of points array concatenation is great, but it does not scale well.
>
> paulv

>
> [*] When I had to read many, many of the same type of files I could walk down the street
> to get a coffee before it had finished.
>
> --
> Paul van Delst Ride lots.
> CIMSS @ NOAA/NCEP/EMC Eddy Merckx
> Ph: (301)763-8000 x7748
> Fax:(301)763-8545
>
