Subject: Re: Assignment Time for a 3d Variable
Posted by Antonio Santiago on Wed, 23 Nov 2005 12:46:41 GMT
View Forum Message <> Reply to Message

Nuno Oliveira wrote:
> I was making a routine that was doing an intense assignment in one of
> the three directions possible, according to an option (either the first,
> second or third dimension). I noticed that when I was doing it in the
> first direction I took MUCH more time than in the other two directions.
>
> Anyone has a clue for why does this happen? And anyone knows a way that
> can make execution time similar? (Making the others to wait is not a
> valid answer, ;))
>
> Here is the code for checking the execution times:
>
```
> time = systime(1)
> for k = 0, 99 do $
>    temp = vol[k,*,*]
> print, 'execution time (for x axis): '+STRING(systime(1) - time)
>
> time = systime(1)
> for k = 0, 99 do $
>    temp = vol[*,k,*]
> print, 'execution time (for y axis): '+STRING(systime(1) - time)
>
> time = systime(1)
> for k = 0, 99 do $
>    temp = vol[*,k,*]
> print, 'execution time (for z axis): '+STRING(systime(1) - time)
```
>
> And this is what I get:
>
```
> execution time (for x axis):      0.24305296
> execution time (for y axis):    0.0063638687
> execution time (for z axis):    0.0065510273
>
```

I suposse it depends on the way IDL (and internally C) stores arrays.
http://www.ibiblio.org/pub/languages/fortran/ch2-6.html

--

Antonio Santiago Pï¿½rez
(email: santiago<<at>>grahi.upc.edu   )
(www: http://www.grahi.upc.edu/santiago)
(www: http://asantiago.blogsite.org   )

GRAHI - Grup de Recerca Aplicada en Hidrometeorologia
Universitat Politï¿½cnica de Catalunya

Subject: Re: Assignment Time for a 3d Variable
Posted by David Fanning on Wed, 23 Nov 2005 15:57:57 GMT
View Forum Message <> Reply to Message

Nuno Oliveira writes:

> I was making a routine that was doing an intense assignment in one of
> the three directions possible, according to an option (either the first,
> second or third dimension). I noticed that when I was doing it in the
> first direction I took MUCH more time than in the other two directions.
>
> Anyone has a clue for why does this happen? And anyone knows a way that
> can make execution time similar? (Making the others to wait is not a
> valid answer, ;))

The speed differences have to do with how you access different
parts of the array in memory. If the parts you want are contiguous,
then you can get them faster than you can if they are far apart in
memory. (Think how much faster it is to pick up the poker
chips when they are stacked than when they are scattered all
around the table.)

To make these kinds of assignments as fast as possible, use
the TRANSPOSE function to organize the data into the fastest
possible position:

```
  IDL> Help, data
    DATA          BYTE      = Array[3, 227, 149]
  IDL> data = Transpose( Temporary(data), [2,3,1] )
  IDL> Help, data
    DATA          BYTE      = Array[227, 149, 3]
```

Cheers,

David

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

Subject: Re: Assignment Time for a 3d Variable
Posted by David Streutker on Wed, 23 Nov 2005 16:52:46 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
```
>  The speed differences have to do with how you access different
>  parts of the array in memory. If the parts you want are contiguous,
>  then you can get them faster than you can if they are far apart in
>  memory. (Think how much faster it is to pick up the poker
>  chips when they are stacked than when they are scattered all
>  around the table.)
>
>  To make these kinds of assignments as fast as possible, use
>  the TRANSPOSE function to organize the data into the fastest
>  possible position:
>
>     IDL> Help, data
>       DATA           BYTE      = Array[3, 227, 149]
>     IDL> data = Transpose( Temporary(data), [2,3,1] )
>     IDL> Help, data
>       DATA           BYTE      = Array[227, 149, 3]
```

How does one know which is the fastest possible position? Should the largest dimension be first? Nuno's example seems to imply that the first dimension is not the fastest accessed.

-Dave

Subject: Re: Assignment Time for a 3d Variable
Posted by Antonio Santiago on Wed, 23 Nov 2005 17:51:20 GMT
View Forum Message <> Reply to Message

On Wed, 2005-11-23 at 08:52 -0800, David Streutker wrote:
```
>  David Fanning wrote:
>>  The speed differences have to do with how you access different
>>  parts of the array in memory. If the parts you want are contiguous,
>>  then you can get them faster than you can if they are far apart in
>>  memory. (Think how much faster it is to pick up the poker
>>  chips when they are stacked than when they are scattered all
>>  around the table.)
>>
>>  To make these kinds of assignments as fast as possible, use
>>  the TRANSPOSE function to organize the data into the fastest
>>  possible position:
>>
>>     IDL> Help, data
>>       DATA           BYTE      = Array[3, 227, 149]
>>     IDL> data = Transpose( Temporary(data), [2,3,1] )
>>     IDL> Help, data
>>       DATA            BYTE      = Array[227, 149, 3]
```
>
> How does one know which is the fastest possible position? Should the
> largest dimension be first? Nuno's example seems to imply that the
> first dimension is not the fastest accessed.
>
> -Dave
>

Also I note that it depends on the language IDL is implemented. By
default all ANSI C compilers use row major mode.
See the other post for the link to an example :)

--

Subject: Re: Assignment Time for a 3d Variable
Posted by K. Bowman on Wed, 23 Nov 2005 18:16:53 GMT
View Forum Message <> Reply to Message

In article <1132768280.23827.0.camel@localhost.localdomain>,
Antonio Santiago <santiago@grahi.upc.edu> wrote:

> Also I note that it depends on the language IDL is implemented. By
> default all ANSI C compilers use row major mode.
> See the other post for the link to an example :)

IDL uses the Fortran convention for storage of arrays, that is, the first
subscript varies fastest.

"The fact that the elements of the first dimension are contiguous means that the
elements of each row of an image array are contiguous. This is the order
expected by most graphics hardware, providing an efficiency advantage for
languages that naturally store data that way. Also, this ordering minimizes
virtual memory overhead, since images are accessed linearly."

Ken Bowman

Subject: Re: Assignment Time for a 3d Variable
Posted by James Kuyper on Wed, 23 Nov 2005 18:35:21 GMT
View Forum Message <> Reply to Message

David Streutker wrote:
> David Fanning wrote:

```
>>  The speed differences have to do with how you access different
>>  parts of the array in memory. If the parts you want are contiguous,
>>  then you can get them faster than you can if they are far apart in
>>  memory. (Think how much faster it is to pick up the poker
>>  chips when they are stacked than when they are scattered all
>>  around the table.)
>>
>>  To make these kinds of assignments as fast as possible, use
>>  the TRANSPOSE function to organize the data into the fastest
>>  possible position:
>>
>>     IDL> Help, data
>>       DATA           BYTE      = Array[3, 227, 149]
>>     IDL> data = Transpose( Temporary(data), [2,3,1] )
>>     IDL> Help, data
>>       DATA           BYTE      = Array[227, 149, 3]
>
>  How does one know which is the fastest possible position?  Should the
>  largest dimension be first?  Nuno's example seems to imply that the
>  first dimension is not the fastest accessed.
```

Note: Nuno's third case was identical to his second one; the time variation must have been random. With the third case corrected to:

```
for k = 0, 99 do $
        temp = vol[*,*,k]
print, 'execution time (for z axis): '+STRING(systime(1) - time)
```

I get:

```
execution time (for x axis):      0.18765187
execution time (for y axis):     0.038336992
execution time (for z axis):     0.031430006
```

If you're accessing consecutive positions along one dimension of an object, then the first dimension is in fact the fastest one, because those consecutive positions are physically adjacent to each other.. That corresponds the fastest case, the third one. The slowest case was when only one position along the first dimension was accessed for all possible values along the other two dimensions.

Subject: Re: Assignment Time for a 3d Variable
Posted by David Fanning on Wed, 23 Nov 2005 19:09:20 GMT

View Forum Message <> Reply to Message

David Streutker writes:

> How does one know which is the fastest possible position? Should the
> largest dimension be first? Nuno's example seems to imply that the
> first dimension is not the fastest accessed.

Well, uh, I'd run a couple of timing tests, I guess.
I imagine it will depend on exactly what it is you are
doing. :-)

Cheers,

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

Subject: Re: Assignment Time for a 3d Variable
Posted by Foldy Lajos on Wed, 23 Nov 2005 19:13:37 GMT
View Forum Message <> Reply to Message

Hi,

Fortran is column major, IDL and C are row major. In IDL (unlike C or
Fortran) the first subscript is column, not row.

It is best not to refer to Fortran or C, IDL is like IDL :-)

regards,
lajos

On Wed, 23 Nov 2005, Kenneth Bowman wrote:

> In article <1132768280.23827.0.camel@localhost.localdomain>,
> Antonio Santiago <santiago@grahi.upc.edu> wrote:
>
>> Also I note that it depends on the language IDL is implemented. By
>> default all ANSI C compilers use row major mode.
>> See the other post for the link to an example :)
>
> IDL uses the Fortran convention for storage of arrays, that is, the first
> subscript varies fastest.
>
> "The fact that the elements of the first dimension are contiguous means that the
> elements of each row of an image array are contiguous. This is the order
> expected by most graphics hardware, providing an efficiency advantage for

> languages that naturally store data that way. Also, this ordering minimizes
> virtual memory overhead, since images are accessed linearly."
>
> Ken Bowman
>

Subject: Re: Assignment Time for a 3d Variable
Posted by Mark Hadfield on Wed, 23 Nov 2005 20:17:04 GMT

View Forum Message <> Reply to Message

Fï¿½ldy Lajos wrote:
> Hi,
>
> Fortran is column major, IDL and C are row major. In IDL (unlike C or
> Fortran) the first subscript is column, not row.
>
> It is best not to refer to Fortran or C, IDL is like IDL :-)

Dragging in "rows" and "columns" here is a distraction. They relate
(presumably) to how you print an array or possibly to how you interpret
the array dimensions in matrix operations. I am not really a
matrix-oriented guy and that is why I find Matlab so painful. It also
may be why when I hear people use the terms row-major and column-major I
put my fingers in my ears and go "la la la" until the noise stops.

IDL is like Fortran in that the left-most index refers to the dimension
that varies fastest in memory. Ie. if we have a = fltarr(3,2) then the
order of the elements in memory is...

```
  a[0,0] a[1,0] a[2,0] a[0,1] a[1,1] a[2,1]
```

So it generally makes sense to loop over the left-most index (inner
dimension)

```
  a = fltarr(m, n)
  for j=0,n-1 do begin
    for i=0,m-1 do begin
      ; Do something with a[i,j]
    endfor
  endfor
```

This way you're stepping thru memory from right to left.

Previous messages have suggested that because IDL is implemented in C it
must use C's convention. Not so.

--

Mark Hadfield "Kei puwaha te tai nei, Hoea tahi tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)