Subject: Re: temporary() pitfall
Posted by landsman on Mon, 18 Dec 2000 23:00:23 GMT
View Forum Message <> Reply to Message

In article <3A3E0D3C.23741E8E@fz-juelich.de>,
 j.c.van.gorkom@fz-juelich.de wrote:
> I use the temporary(MyArray) function to conserve memory when doing
> operations on large arrays. This works fine, except if there is still
> not enough memory to do the operation. Then IDL stops execution, tells
> me there is not enough memory, and MyArray is undefined. IDL follows
the
> documentation here, but I lose my array contents!
>
> Does anyone know of a way
> - to test in advance whether the operation is going to fail, or
> - to recover the contents of my original array?
>
> The operation I do varies, today I was trying to do
> MyArray = transpose(temporary(MyArray))

Jaco,

Well, I think this is a case where you can't get something for nothing.
The memory that you save with TEMPORARY() comes at the cost of losing
the original array contents. If you are worried about losing the
result of a long computation because of hitting a memory limit, then I
would SAVE the array to disk first. (I find that programs that use a
lot of TEMPORARY calls are also difficult to debug.)

If you have IDL V5.4 and are transposing prior to a matrix
multiplication, you might look at the /ATRANSPOSE and /BTRANSPOSE
keywords to MATRIX_MULTIPLY which do an implicit transpose to save
memory. (The transpose is done simultaneously with the matrix
mutliplication.)

Finally, I'd assert (without complete confidence) that there is no sense
in using the TEMPORARY function unless you have at least two variables
or constants on the right hand side of the "=" sign. One way to see
this is to use the /HIGHWATER keyword to the MEMORY function introduced
in V5.4. This returns the maximum amount of dynamic memory used since
the last time the MEMORY function or HELP,/MEMORY was called. (Users
without 5.4 can look at the MAX value in a HELP,/MEMORY call.)

First, look at a case where a call to TEMPORARY() really is useful.

```
IDL>print,!VERSION
  sparc sunos unix 5.4 Sep 25 2000      64      64}
```

```
IDL> a = lonarr(2048,2048) & tmp =memory() &  a = a+1  &
print,memory(/high)
   50834756
IDL> a = lonarr(2048,2048) & tmp =memory() & a = temporary(a)+1
IDL> print,memory(/high)
   34057465
```

So, here a = a+1 requires 50 Mb of memory, but a = temporary(a)+1 only
requires 34 Mb of memory.

But as shown below, there is no advantage of writing
a=transpose(temporary(a))
rather than simply a = transpose(a).

```
IDL> a = lonarr(2048,2048) & tmp =memory() &  a = transpose(a)
IDL> print,memory(/high)
   50834836
IDL> a = lonarr(2048,2048)& tmp =memory() &  a = transpose(temporary(a))
IDL> print,memory(/high)
   50834836
```

One reason I don't have complete confidence in this result is that at
least one RSI-supplied procedure (laguerre.pro) includes the statement

```
   xPrec = FLOAT(TEMPORARY(xPrec))
```

which I'd say is a useless use of TEMPORARY().

Wayne Landsman   landsman@mpb.gsfc.nasa.gov

Sent via Deja.com
http://www.deja.com/

Subject: Re: temporary() pitfall
Posted by davidf on Mon, 18 Dec 2000 23:20:06 GMT

View Forum Message <> Reply to Message

Wayne Landsman (landsman@my-deja.com) writes:

> One reason I don't have complete confidence in this result is that at
> least one RSI-supplied procedure (laguerre.pro) includes the statement
>
>   xPrec = FLOAT(TEMPORARY(xPrec))
>
> which I'd say is a useless use of TEMPORARY().

Uh, let's not get started on the quality of RSI-supplied
IDL code. :-(

Cheers,

David

P.S. Let's just say generosity is a hallmark of
the Holiday Season.

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: temporary() pitfall
Posted by Jaco van Gorkom on Tue, 19 Dec 2000 17:42:11 GMT
View Forum Message <> Reply to Message

Thanks, Wayne. I gained some new insights here.

> The memory that you save with TEMPORARY() comes at the cost of losing
> the original array contents. If you are worried about losing the
> result of a long computation because of hitting a memory limit, then I
> would SAVE the array to disk first. (I find that programs that use a
> lot of TEMPORARY calls are also difficult to debug.)

I agree that losing the original contents is the price that I was
willing to pay. I guess I was just hoping that someone here would come
up with another secret and magical keyword to ROUTINE_NAMES(), to
recover that which seems lost forever. Always keep hoping for a
miracle...

SAVEing to disk is of course the best option, but I did not expect
TRANSPOSE to need a lot of memory. Especially not since it is mentioned
as an example of smart memory use in the Online Help (under "Virtual
Memory"). Now that I tried your examples, I realize that I should not
have used A = TRANSPOSE(TEMPORARY(A)), but something like A =
TRANSPOSE(A, /INPLACE), using an (IDL5.5???) /INPLACE or /OVERWRITE or
/NOCOPY keyword to prohibit TRANSPOSE from copying the whole array. Bad
luck for now.

Thanks for pointing out the behaviour of the MAX value in HELP,/MEMORY ;
that seems to be a good method for testing the memory expense of certain

operations. (I have not installed IDL5.4 yet, because I really do not
want to miss the colored listings in idlde. And yes, I have *tried* to
install xemacs for IDLWAVE. But let's not get started on installing
under Unix again...)

Jaco

Jaco van Gorkom gorkom@rijnh.nl
FOM-Instituut voor Plasmafysica Rijnhuizen

Subject: Re: temporary() pitfall
Posted by Paul van Delst on Tue, 19 Dec 2000 19:58:44 GMT
View Forum Message <> Reply to Message

Jaco van Gorkom wrote:
>
> Thanks, Wayne. I gained some new insights here.
>
>> The memory that you save with TEMPORARY() comes at the cost of losing
>> the original array contents. If you are worried about losing the
>> result of a long computation because of hitting a memory limit, then I
>> would SAVE the array to disk first. (I find that programs that use a
>> lot of TEMPORARY calls are also difficult to debug.)
>
> I agree that losing the original contents is the price that I was
> willing to pay. I guess I was just hoping that someone here would come
> up with another secret and magical keyword to ROUTINE_NAMES(), to
> recover that which seems lost forever. Always keep hoping for a
> miracle...
>
> SAVEing to disk is of course the best option,

Hmm, why not re-design the code to work in a smaller memory footprint? (E.g. using smarter, memory
efficient algorithms for doing linear algebra based on the type of matrix; sparse, banded, dense,
etc.) The up front cost will be high (wrt time at least), but at least you'll know the code has a
better chance of working when your dataset/data flow increases 100-fold.

jsut me musing and mucking about.

paulv

--
Paul van Delst Ph: (301) 763-8000 x7274
CIMSS @ NOAA/NCEP Fax: (301) 763-8545
Rm.207, 5200 Auth Rd. Email: pvandelst@ncep.noaa.gov

Camp Springs MD 20746

Subject: Re: temporary() pitfall
Posted by Jaco van Gorkom on Thu, 21 Dec 2000 17:26:21 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> Jaco van Gorkom wrote:
>>
>>> The memory that you save with TEMPORARY() comes at the cost of losing
>>> the original array contents.   If you are worried about losing the
>>> result of a long computation because of hitting a memory limit, then I
>>> would SAVE the array to disk first. (I find that programs that use a
>>> lot of TEMPORARY calls are also difficult to debug.)
>>>
>> SAVEing to disk is of course the best option,
>
> Hmm, why not re-design the code to work in a smaller memory footprint? (E.g. using smarter, memory
> efficient algorithms for doing linear algebra based on the type of matrix; sparse, banded, dense,
> etc.) The up front cost will be high (wrt time at least), but at least you'll know the code has a
> better chance of working when your dataset/data flow increases 100-fold.
>

I agree. I hope to get the code geared up before Christmas. I
encountered the memory problems while using IDL interactively, trying to
take full advantage of IDL's near-zero up front cost. Maybe I should
code a little widget to keep track of memory usage all the time, which
starts beeping and flashing when I approach the limit. Anything to make
interactive life easier in a world full of object graphics and other
enhancements.

Jaco