Subject: Re: Array has too many elements?
Posted by Jonathan Greenberg on Fri, 19 Dec 2003 08:51:29 GMT
View Forum Message <> Reply to Message

Followup question -- I've now come across a lot of discussion about windows
boxes and memory problems under IDL (David Fanning, thats a terrific website
you have (http://www.dfanning.com/)-- wish I had found it a few weeks ago!).
Do these problems simply not exist on UNIX boxes (given ample amount of RAM
+ pagefile space)? I reprogrammed a slow but memory conservative version of
this algorithm I'm working on using a bunch of for-next loops -- after all
the discussion I've been seeing about how for-next loops stink, I recoded
the algorithm using mostly matrix calls -- but of course I need more memory
to do these, hence the array has too many elements error. I'm tempted to
switch over to a slower UNIX box if this memory problem won't show up...
thoughts?

--j

"Jonathan Greenberg" <greenberg@ucdavis.edu> wrote in message
news:amyEb.209$rX6.10@newssvr29.news.prodigy.com...
> Is there some intrinsic limitation in how big an array can get in IDL?
The
> one in question was a fltarr of 170 x ~ 4,000,000 which gave me the "Array
> has too many elements." error. I have about 2gb of memory on my machine,
> and a 2 gb pagefile. Did I simply run out of memory, or is there
something
> else going on here?
>
> --j
>
>

Subject: Re: Array has too many elements?
Posted by Jamie on Fri, 19 Dec 2003 09:23:17 GMT
View Forum Message <> Reply to Message

Take a look at the idl/external/export.h file that came with IDL as this
defines most of the limits. You, of course, cannot change them but it is
helpful to know what they are. In particular, getting beyond the 32-bit
address limit is problematic and very OS dependent. Unless I missed
something in version 6, IDL is still compiled as a 32-bit application.

See:
http://www.rsinc.com/services/techtip.asp?ttid=2597
http://www.rsinc.com/services/techtip.asp?ttid=3617

Or do a search for 'memory' in:

http://www.rsinc.com/services/search.asp

Jamie

On Fri, 19 Dec 2003, Jonathan Greenberg wrote:

> Followup question -- I've now come across a lot of discussion about windows
> boxes and memory problems under IDL (David Fanning, thats a terrific website
> you have (http://www.dfanning.com/)-- wish I had found it a few weeks ago!).
> Do these problems simply not exist on UNIX boxes (given ample amount of RAM
> + pagefile space)? I reprogrammed a slow but memory conservative version of
> this algorithm I'm working on using a bunch of for-next loops -- after all
> the discussion I've been seeing about how for-next loops stink, I recoded
> the algorithm using mostly matrix calls -- but of course I need more memory
> to do these, hence the array has too many elements error. I'm tempted to
> switch over to a slower UNIX box if this memory problem won't show up...
> thoughts?
>
> --j
>
> "Jonathan Greenberg" <greenberg@ucdavis.edu> wrote in message
> news:amyEb.209$rX6.10@newssvr29.news.prodigy.com...
>> Is there some intrinsic limitation in how big an array can get in IDL?
> The
>> one in question was a fltarr of 170 x ~ 4,000,000 which gave me the "Array
>> has too many elements." error. I have about 2gb of memory on my machine,
>> and a 2 gb pagefile. Did I simply run out of memory, or is there
> something
>> else going on here?
>>
>> --j

Subject: Re: Array has too many elements?
Posted by Bringfried Stecklum on Fri, 19 Dec 2003 11:00:55 GMT
View Forum Message <> Reply to Message

Jonathan Greenberg wrote:
> Followup question -- I've now come across a lot of discussion about windows
> boxes and memory problems under IDL (David Fanning, thats a terrific website
> you have (http://www.dfanning.com/)-- wish I had found it a few weeks ago!).
> Do these problems simply not exist on UNIX boxes (given ample amount of RAM
> + pagefile space)? I reprogrammed a slow but memory conservative version of
> this algorithm I'm working on using a bunch of for-next loops -- after all
> the discussion I've been seeing about how for-next loops stink, I recoded
> the algorithm using mostly matrix calls -- but of course I need more memory
> to do these, hence the array has too many elements error. I'm tempted to
> switch over to a slower UNIX box if this memory problem won't show up...

> thoughts?
>
> --j
>
You need a 64 bit CPU to overcome the memory problem (or more accurately, to
shift the barrier beyond your current needs). The ~1.5GB limit in memory and
the maximum amount of ~2^31 array elements result from the 32 bit architecture
(irrespective of running windows or linux).

regards,

 Bringfried Stecklum

Subject: Re: Array has too many elements?
Posted by Michael Wallace on Fri, 19 Dec 2003 16:32:31 GMT
View Forum Message <> Reply to Message

According to the "What's New In IDL 5.4" manual, RSI added large file (
 > 2.1GB) support for Windows. Apparently, you can use the 64-bit
integer data type to read and write data from files. However, there
aren't any examples and I've never tried it myself since I avoid Windows
like the plague.

HTH,

Mike W

Jamie wrote:
> Take a look at the idl/external/export.h file that came with IDL as this
> defines most of the limits. You, of course, cannot change them but it is
> helpful to know what they are. In particular, getting beyond the 32-bit
> address limit is problematic and very OS dependent. Unless I missed
> something in version 6, IDL is still compiled as a 32-bit application.
>
> See:
> http://www.rsinc.com/services/techtip.asp?ttid=2597
> http://www.rsinc.com/services/techtip.asp?ttid=3617
>
> Or do a search for 'memory' in:
> http://www.rsinc.com/services/search.asp
>
> Jamie
>
> On Fri, 19 Dec 2003, Jonathan Greenberg wrote:
>
>
>> Followup question -- I've now come across a lot of discussion about windows

>> boxes and memory problems under IDL (David Fanning, thats a terrific website
>> you have (http://www.dfanning.com/)-- wish I had found it a few weeks ago!).
>> Do these problems simply not exist on UNIX boxes (given ample amount of RAM
>> + pagefile space)? I reprogrammed a slow but memory conservative version of
>> this algorithm I'm working on using a bunch of for-next loops -- after all
>> the discussion I've been seeing about how for-next loops stink, I recoded
>> the algorithm using mostly matrix calls -- but of course I need more memory
>> to do these, hence the array has too many elements error. I'm tempted to
>> switch over to a slower UNIX box if this memory problem won't show up...
>> thoughts?
>>
>> --j
>>
>> "Jonathan Greenberg" <greenberg@ucdavis.edu> wrote in message
>> news:amyEb.209$rX6.10@newssvr29.news.prodigy.com...
>>
>>> Is there some intrinsic limitation in how big an array can get in IDL?
>>
>> The
>>
>>> one in question was a fltarr of 170 x ~ 4,000,000 which gave me the "Array
>>> has too many elements." error. I have about 2gb of memory on my machine,
>>> and a 2 gb pagefile. Did I simply run out of memory, or is there
>>
>> something
>>
>>> else going on here?
>>>
>>> --j
>
>

Subject: Re: Array has too many elements?
Posted by Michael Wallace on Fri, 19 Dec 2003 16:46:06 GMT
View Forum Message <> Reply to Message

Silly me. I misread the earlier message. I was thinking of large file
support rather than RAM. Oops.

The Windows version of IDL is still compiled as a 32-bit application.
It's only been recently that Windows has even produced a 64-bit version
of their operating systems, whereas Sun and other Unixes had 64-bit
operating systems years ago. So, it will be a while before RSI catches
up and produces a 64-bit Windows version.

Michael Wallace wrote:
> According to the "What's New In IDL 5.4" manual, RSI added large file (

>> 2.1GB) support for Windows. Apparently, you can use the 64-bit
> integer data type to read and write data from files. However, there
> aren't any examples and I've never tried it myself since I avoid Windows
> like the plague.
>

Subject: Re: Array has too many elements?
Posted by Jonathan Greenberg on Fri, 19 Dec 2003 19:54:46 GMT
View Forum Message <> Reply to Message

So, any suggestions for the best way of getting around these limitations (I
mean, without having to buy a 64-bit machine) -- how about chopping the
array up into smaller blocks and performing for-next loops -- processing
part of the array, writing the results, and then processing the next part?
Are there better ways than this?

--j

"Bringfried Stecklum" <stecklum@tls-tautenburg.de> wrote in message
news:brufv7$qgm$1@fsuj29.rz.uni-jena.de...
> Jonathan Greenberg wrote:
>> Followup question -- I've now come across a lot of discussion about
windows
>> boxes and memory problems under IDL (David Fanning, thats a terrific
website
>> you have (http://www.dfanning.com/)-- wish I had found it a few weeks
ago!).
>> Do these problems simply not exist on UNIX boxes (given ample amount of
RAM
>> + pagefile space)? I reprogrammed a slow but memory conservative
version of
>> this algorithm I'm working on using a bunch of for-next loops -- after
all
>> the discussion I've been seeing about how for-next loops stink, I
recoded
>> the algorithm using mostly matrix calls -- but of course I need more
memory
>> to do these, hence the array has too many elements error. I'm tempted
to
>> switch over to a slower UNIX box if this memory problem won't show up...
>> thoughts?
>>
>> --j
>>
> You need a 64 bit CPU to overcome the memory problem (or more accurately,
to

> shift the barrier beyond your current needs). The ~1.5GB limit in memory
and
> the maximum amount of ~2^31 array elements result from the 32 bit
architecture
> (irrespective of running windows or linux).
>
> regards,
>
> Bringfried Stecklum
>

Subject: Re: Array has too many elements?
Posted by Jamie on Mon, 22 Dec 2003 18:32:30 GMT
View Forum Message <> Reply to Message

On Fri, 19 Dec 2003, Jonathan Greenberg wrote:

> So, any suggestions for the best way of getting around these limitations (I
> mean, without having to buy a 64-bit machine) -- how about chopping the
> array up into smaller blocks and performing for-next loops -- processing
> part of the array, writing the results, and then processing the next part?
> Are there better ways than this?

Nope. Frankly, once you start working with arrays that consume 1GB of
memory, you are in for a whole world of trouble. IDL is a flexible,
non-compiled language which means that commands are expanded into a
working stack where copies are often made. Working with big arrays also
means becoming proficeint in using the NOZERO keyword, the
REPLICATE_INPLACE procedure, the NO_COPY keyword, and the TEMPORARY
function. You haven't really told us what exactly you are trying to do...
I hope that you don't have too many zeros in your arrays ;)

As far as I know, the array size limit in IDL on *all* platforms is still
2^31-1 (2 GB). Overcoming this is more-or-less impossible even with a 64
-bit machine. In short, you need to break up large arrays and be careful
with your indexing. The best case would be if you can effectively reduce
the volume of the data as you loop.

Keep in mind that IDL stands for "interactive data language." While, it
is a capable programming environment for visualizing data, working with
arrays that consume 1-2 GB is not quite mainstream yet. In another 2
years, this probably won't be such an issue...

Jamie

Subject: Re: Array has too many elements?
Posted by Jonathan Greenberg on Mon, 22 Dec 2003 21:35:55 GMT
View Forum Message <> Reply to Message

I'm working with a fuzzy set classifier I found an article on that
subdivides a dataspace into subspaces of smaller and smaller divisions. So,
for instance, if you have 2 attributes and 2 divisions per attribute, you
have an array that has 2^2 elements. Most datasets really need about 25
subdivisions per attribute, and, say, 6 attributes to start getting good
classification, so this problem blows up VERY quickly (e.g. 25^6 elements
for the problem I just described). One of the issues is that I need to
extract a maximum value from the 25^6 array, which, it now looks like, i'll
have to do in stages (e.g. subdivide the array into fixed size subsets,
check each subset for max value and write to a new array, and then determine
the max value for this new array). Doable, but obviously involves rewriting
a lot of code.

This, by the way, is why I brought up the supercomputer thread --
manipulating arrays (even if I do get the programming bugs sorted out) of
this size will be silly to do with a desktop PC. I hope IDL modifies how
they deal with large arrays in the future -- since remote sensing images are
getting bigger as the spatial, extent and spectral resolution gets higher,
this problem is only going to get worse.

--j

"Jamie" <jamiedotwheeleratoxacuk@dummy.com> wrote in message
news:Pine.LNX.4.44.0312221805210.13640-100000@moriarty.atm.o x.ac.uk...
>
> On Fri, 19 Dec 2003, Jonathan Greenberg wrote:
>
>> So, any suggestions for the best way of getting around these limitations
(I
>> mean, without having to buy a 64-bit machine) -- how about chopping the
>> array up into smaller blocks and performing for-next loops -- processing
>> part of the array, writing the results, and then processing the next
part?
>> Are there better ways than this?
>
> Nope. Frankly, once you start working with arrays that consume 1GB of
> memory, you are in for a whole world of trouble. IDL is a flexible,
> non-compiled language which means that commands are expanded into a
> working stack where copies are often made. Working with big arrays also
> means becoming proficeint in using the NOZERO keyword, the
> REPLICATE_INPLACE procedure, the NO_COPY keyword, and the TEMPORARY
> function. You haven't really told us what exactly you are trying to do...
> I hope that you don't have too many zeros in your arrays ;)
>

> As far as I know, the array size limit in IDL on *all* platforms is still
> 2^31-1 (2 GB). Overcoming this is more-or-less impossible even with a 64
> -bit machine. In short, you need to break up large arrays and be careful
> with your indexing. The best case would be if you can effectively reduce
> the volume of the data as you loop.
>
> Keep in mind that IDL stands for "interactive data language." While, it
> is a capable programming environment for visualizing data, working with
> arrays that consume 1-2 GB is not quite mainstream yet. In another 2
> years, this probably won't be such an issue...
>
> Jamie
>

Subject: Re: Array has too many elements?
Posted by JD Smith on Mon, 22 Dec 2003 22:01:30 GMT

View Forum Message <> Reply to Message

On Mon, 22 Dec 2003 14:35:55 -0700, Jonathan Greenberg wrote:

> I'm working with a fuzzy set classifier I found an article on that
> subdivides a dataspace into subspaces of smaller and smaller divisions.
> So, for instance, if you have 2 attributes and 2 divisions per
> attribute, you have an array that has 2^2 elements. Most datasets
> really need about 25 subdivisions per attribute, and, say, 6 attributes
> to start getting good classification, so this problem blows up VERY
> quickly (e.g. 25^6 elements for the problem I just described). One of
> the issues is that I need to extract a maximum value from the 25^6
> array, which, it now looks like, i'll have to do in stages (e.g.
> subdivide the array into fixed size subsets, check each subset for max
> value and write to a new array, and then determine the max value for
> this new array). Doable, but obviously involves rewriting a lot of
> code.
>
> This, by the way, is why I brought up the supercomputer thread --
> manipulating arrays (even if I do get the programming bugs sorted out)
> of this size will be silly to do with a desktop PC. I hope IDL modifies
> how they deal with large arrays in the future -- since remote sensing
> images are getting bigger as the spatial, extent and spectral resolution
> gets higher, this problem is only going to get worse.

Hmmm, I'm not sure about your desktop PC, but that's only a .9GB
array... nothing to get bothered by:

```
IDL> big=randomu(sd,replicate(25L,6)) & print,size(big,/DIMENSIONS) & t=systime(1) &
m=max(big,pos) & print,systime(1)-t & print,m,pos
```

```
          25          25          25          25          25          25
       1.7971461
     1.000000    68392329
```

on a two-year old dual processor Athlon system with 1GB RAM. Give me
a fast new pair of processors and 4GB of RAM, and this would be a
cake-walk, as long as you're careful to keep extra copies of the array
to a minimum. If you're desperate for speed, MAX is easy to write in
C: just read in chunks at a time about the size of your memory and
loop through.

JD