Subject: working with very large data sets
Posted by Steve Carothers on Thu, 18 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

I am working with 71 Mb data file on UNIX server that has 256 Mb RAM and 500 Mb of virtual memory. I'm not doing much data manipulation before I plot the data, but it doesn't take much manipulation to exceed the memory allocation. I understand the benefits of chunking up the data, but I would really like to keep all the data together for plotting purposes. I think my PV-Wave script could run properly if I can figure out how to minimize or eliminate memory fragmentation. When I'm done with a variable I set it equal to 0 to free up the memory. However, if I understand the manual correctly, this will not free up contiguous memory, which is what I need. Delstruct and delvar might help me but they can't be used inside a script, only at the prompt. I have a feeling I'll be forced to chunk up the data.

Also, is there a way to remove a set of records from an array of structures if the records to delete are known without using the "where" command and without creating a tempory variable in the memory?

Any advice would be appreciated.

Steve

Subject: Re: working with very large data sets
Posted by David Foster on Thu, 25 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Steve Carothers wrote:
>
> I am working with 71 Mb data file on UNIX server that has 256 Mb RAM and 500
> Mb of virtual memory. I'm not doing much data manipulation before I plot
> the data, but it doesn't take much manipulation to exceed the memory
> allocation. I understand the benefits of chunking up the data, but I would
> really like to keep all the data together for plotting purposes. I think my
> PV-Wave script could run properly if I can figure out how to minimize or
> eliminate memory fragmentation. When I'm done with a variable I set it
> equal to 0 to free up the memory. However, if I understand the manual
> correctly, this will not free up contiguous memory, which is what I need.
> Delstruct and delvar might help me but they can't be used inside a script,
> only at the prompt. I have a feeling I'll be forced to chunk up the data.

Steve -

What manipulations are you doing?! 256MB certainly seems like it should
be
enough memory. Try setting the "limit" parameter. Also, you might want

to
download TOP, a utility that shows you how much memory and cpu usage
each
process is using...may help you trace which steps are increasing your
memory usage, as the top display is updated every 5 seconds. You can get
top from:

 ftp.groupsys.com:/pub/top/top-3.5beta7.tar.gz

>
> Also, is there a way to remove a set of records from an array of structures
> if the records to delete are known without using the "where" command and
> without creating a tempory variable in the memory?

If you mean that you would like to remove elements of your array of
structures,
then you can do this simply with something like:

```
 ToDel = [3,22,89]                    ; Known indices to remove
 Indices = lindgen(n_elements(Array))
 NewInd = SetDifference(Indices, ToDel)     ; SetDifference() included
below

 Array = (temporary(Array))[NewInd]
```

If the memory overhead for this method is too high, you might want to
consider
creating a linked list of structures instead of a simple array; this
allows you
to delete/add nodes easily, with no memory overhead.

Dave
--

```
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~
    David S. Foster         Univ. of California, San Diego
     Programmer/Analyst     Brain Image Analysis Laboratory
     foster@bial1.ucsd.edu  Department of Psychiatry
     (619) 622-5892         8950 Via La Jolla Drive, Suite 2240
                            La Jolla, CA  92037
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~
```

```
; ________________________________________________________________ _____
;
; SETARRAY_UTILS.PRO  [RSI]  9-04-97
;
; Routines posted on newsgroup by RSI. SetIntersection() is much
; faster than Find_Elements(), but it returns the elements
```

```
; themselves, not the indices. Also, it ignores duplicate elements.
;
;
; Set operators.  Union, Intersection, and Difference (i.e. return
; members of A that are not in B.)
;
; These functions operate on arrays of positive integers, which need
; not be sorted.  Duplicate elements are ignored, as they have no
; effect on the result.
;
; The empty set is denoted by an array with the first element equal to -1.
;
; These functions will not be efficient on sparse sets with wide
; ranges, as they trade memory for efficiency.  The HISTOGRAM function
; is used, which creates arrays of size equal to the range of the
; resulting set.

; For example:
;   a = [2,4,6,8]
;   b = [6,1,3,2]
; SetIntersection(a,b) = [ 2, 6]       ; Common elements
; SetUnion(a,b) = [ 1, 2, 3, 4, 6, 8]  ; Elements in either set
; SetDifference(a,b) = [ 4, 8]         ; Elements in A but not in B
; SetIntersection(a,[3,5,7]) = -1 = Null Set

 ;---------------------------------------------------------- --------
FUNCTION SetUnion, a, b
if a[0] lt 0 then return, b    ;A union NULL = a
if b[0] lt 0 then return, a    ;B union NULL = b
return, where(histogram([a,b], OMIN = omin)) + omin ;Return combined set
end


 ;---------------------------------------------------------- --------
FUNCTION SetIntersection, a, b

minab = min(a, MAX=maxa) > min(b, MAX=maxb) ;Only need intersection of ranges
maxab = maxa < maxb

;If either set is empty, or their ranges don't intersect: result = NULL.
if maxab lt minab or maxab lt 0 then return, -1

r = where((histogram(a, MIN=minab, MAX=maxab) ne 0) and  $
         (histogram(b, MIN=minab, MAX=maxab) ne 0), count)
if count eq 0 then return, -1 else return, r + minab
end
```

```
 ;------------------------------------------------------------ --------
FUNCTION SetDifference, a, b  ; = a and (not b) = elements in A but not in B
mina = min(a, MAX=maxa)
minb = min(b, MAX=maxb)
if (minb gt maxa) or (maxb lt mina) then return, a ;No intersection...
r = where((histogram(a, MIN=mina, MAX=maxa) ne 0) and $
         (histogram(b, MIN=mina, MAX=maxa) eq 0), count)
if count eq 0 then return, -1 else return, r + mina
end


; -------- Message from RSI to NewsGroup -------------------------------
;
; A somewhat belated reply to the numerous postings on finding the
; common elements of vectors:

; > Given vectors of the type...
; >
; > a = [1,2,3,4,5]
; > b = [3,4,5,6,7]
; >
; > What is the most efficient way to determine which values that occur in
; > a also occur in b (i.e., the values [3,4,5] occur in both a and b).
; >

; Below appear three IDL functions that operate on sets represented by
; arrays of positive integers.  The SetIntersection(a,b) function
; returns the common elements, SetUnion(a,b) returns all unique elements
; in both arguments, and SetDifference(a,b) returns the elements
; (members) in a but not in b.

; It is faster than previously published functions, e.g. contain() and
; find_elements().

; Hope this helps,

; Research Systems, Inc.
```

File Attachments

1) setarray_utils.pro, downloaded 98 times