Subject: Re: array concatenation and optimization
Posted by R.G.S. on Wed, 26 Sep 2001 19:22:30 GMT
View Forum Message <> Reply to Message

Sean Raffuse <sean@me.wustl.edu> wrote in message
news:9ot613$3ra$1@newsreader.wustl.edu...
> Hello.
>
> I am trying to read a bunch of data from a file to a structure array. I'm
> not sure many data entries the file will have until I have read it and so
I
> am increasing the size of the structure array after reading each line. I
do
> this by concatenating.
>
> adp_struct_single is the structure as a "scalar"
> adp_struct is the array
>
> I concatenate like so:
> adp_struct =[adp_struct, adp_struct_single]
>
> This is working but it has increased the processing time of my loop by an
> order of magnitude. Is there a better way to do this? Is there a reason
> this is so slow?
>
> Thanks in advance.
>
> -Sean Raffuse

Some ideas:
1) estimate array size from file size (get that from fstat), and
create an appropriately size array.

```
len = fstat(lun).size/nbytesperelement
adp_struct =replicate(adp_struct, len)
```

2) "buffer the concatenate operation
```
 len = 500 ; blocks of 500
adp_struct_block =replicate(adp_struct, len)
while not ( !eof) do begin
  for i = 0,len-1 do begin
    if stillreading then  adp_struct_block(i) = adp_struct_single
  endfor
     adp_struct =[adp_struct, adp_struct_block]
endwhile
```

This is just fakecode, merely illustrating the ideas.

Cheers,
bob

Subject: Re: array concatenation and optimization
Posted by Paul van Delst on Wed, 26 Sep 2001 19:59:17 GMT
View Forum Message <> Reply to Message

Sean Raffuse wrote:
>
> Hello.
>
> I am trying to read a bunch of data from a file to a structure array. I'm
> not sure many data entries the file will have until I have read it and so I
> am increasing the size of the structure array after reading each line.

Count the number of lines first, then create your arrays the required size. (I'm assuming your file is ASCII data).

> This is working but it has increased the processing time of my loop by an
> order of magnitude. Is there a better way to do this? Is there a reason
> this is so slow?

A while back someone else in this newsgroup more knowledgable than me (I think Craig?) explained why this is a bad method when the number of points is very large. Suffice it to say for something like up to maybe a couple hundred points, concatenation is o.k. Any more and you'd be needlessly sucking up CPU cycles shifting/concatenating large blocks of data on the fly.

I changed some of my IDL code that reads a huge binary file from the concat method to estimating the number of points, added some slop and then read it all in a chunk (or in your case, a line) at a time, counting the points as I went. Before returning I simply truncate the array. Sped up my code by at least an order of magnitude.

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: array concatenation and optimization
Posted by Brian Jackel on Wed, 26 Sep 2001 21:29:19 GMT

Hi

This is what I generally do

```
dat= {struct, a:fltarr(3), b:intarr(4), c:0b}   ;example structure
OPENR,lun,filename,/GET_LUN
READU,lun,dat  ;read one record
fs= FSTAT(lun) ;look at current file information
nrec= fs.size/fs.cur_ptr
IF (fs.size GT 100000000L) THEN BEGIN  ;avoid out-of-memory errors
  FREE_LUN,lun
  MESSAGE,'Error- file larger than 100Mbytes, returning'
ENDIF
POINT_LUN,lun,0
data= REPLICATE(dat,nrec)
READU,lun,data   ;read all the data at once
FREE_LUN,lun
```

I'm not sure how well this will work with compressed files.

Brian

Sean Raffuse wrote:
>
> Hello.
>
> I am trying to read a bunch of data from a file to a structure array. I'm
> not sure many data entries the file will have until I have read it and so I
> am increasing the size of the structure array after reading each line. I do
> this by concatenating.
>
> adp_struct_single is the structure as a "scalar"
> adp_struct is the array
>
> I concatenate like so:
> adp_struct =[adp_struct, adp_struct_single]
>
> This is working but it has increased the processing time of my loop by an
> order of magnitude. Is there a better way to do this? Is there a reason
> this is so slow?
>
> Thanks in advance.
>
> -Sean Raffuse

Subject: Re: array concatenation and optimization
Posted by Pavel A. Romashkin on Wed, 26 Sep 2001 21:51:03 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> I changed some of my IDL code that reads a huge binary file from the concat method to
> estimating the number of points, added some slop and then read it all in a chunk (or in your
> case, a line) at a time, counting the points as I went. Before returning I simply truncate the
> array. Sped up my code by at least an order of magnitude.

I have offered this more than once before. It is basically what Paul is suggesting, except I don't read the number of points. I create a buffer (array) with 1000-5000 rows (with any number of columns), then read in a loop, appending the buffer to Result. When READF, Unit, Buffer passes EOF (and causes IO_ERROR, of course, trying to read past the EOF) I redimension the buffer only once, using the value from (FSTATS(unit)).TRANSFER_COUNT. Then, I return to the beginning of last read and read the last buffer correctly.
In fact, you could even ignore redimensioning, close the file and check validity of Buffer and truncate it instead, since Buffer does by then have all data in it from the READF that caused the IO_ERROR.
This eliminates unnecessary reads from the file, as it gets read only once. You don't need to know the number of rows in it.
I never notice this procedure when it reads data on the fly, it is fast. Although I don't seem to work on ASCII files that are over 50 Mb.
In case of numeric data mixed with strings, I found no faster way than to read it all into STRARR and then deaal with it - it is faster to process string array located in RAM than trying to read incrementally from the hard drive. Disk I/O is about the slowest thing you computer can be doing, except maybe accessing a CD-ROM.
Cheers,
Pavel

Subject: Re: array concatenation and optimization
Posted by Andrew Cool on Wed, 26 Sep 2001 23:49:39 GMT
View Forum Message <> Reply to Message

Brian Jackel wrote:
>
> Hi
>
> This is what I generally do
>
> dat= {struct, a:fltarr(3), b:intarr(4), c:0b} ;example structure
> OPENR,lun,filename,/GET_LUN
> READU,lun,dat ;read one record

>   fs= FSTAT(lun) ;look at current file information
>   nrec= fs.size/fs.cur_ptr
>   IF (fs.size GT 100000000L) THEN BEGIN ;avoid out-of-memory errors
>   FREE_LUN,lun
>   MESSAGE,'Error- file larger than 100Mbytes, returning'
>   ENDIF
>   POINT_LUN,lun,0
>   data= REPLICATE(dat,nrec)
>   READU,lun,data   ;read all the data at once
>   FREE_LUN,lun
>
> I'm not sure how well this will work with compressed files.
>
>   Brian
>
> Sean Raffuse wrote:
>>
>> Hello.
>>
>> I am trying to read a bunch of data from a file to a structure array. I'm
>> not sure many data entries the file will have until I have read it and so I
>> am increasing the size of the structure array after reading each line. I do
>> this by concatenating.
>>
>> adp_struct_single   is the structure as a "scalar"
>> adp_struct   is the array
>>
>> I concatenate like so:
>>   adp_struct =[adp_struct, adp_struct_single]
>>
>> This is working but it has increased the processing time of my loop by an
>> order of magnitude. Is there a better way to do this? Is there a reason
>> this is so slow?
>>
>> Thanks in advance.
>>
>> -Sean Raffuse

G'Day,

Under IDL v5.5, the new routine FILE_INFO returns much the same
information as FSTAT, *but* the file doesn't have to be open already!

Nifty.

Andrew Cool

Andrew D. Cool
Electromagnetics & Propagation Group
Surveillance Systems Division
Defence Science & Technology Organisation
PO Box 1500, Salisbury
South Australia 5108

Phone : 061 8 8259 5740 Fax : 061 8 8259 6673
Email : andrew.cool@dsto.defence.gov.au

Subject: Re: array concatenation and optimization
Posted by alt on Thu, 27 Sep 2001 05:05:17 GMT
View Forum Message <> Reply to Message

Our coryphaei are seem to keep silence so I will try to help you
...:-)

> Is there a reason this is so slow?
Because in each concatenation step IDL initializes new variable with
memory freeing, allocation, coping, etc. And it becomes slower and
slower with increasing of array size.
> Is there a better way to do this?
Using temporary() will not help as it seems to.

Very obvious solution is to allocate array of maximum possible size
for data and clip it to filled size at the end.

```
arr = bytarr(10000,/nozero)
i = 0L
while ... do begin
arr[i] = item
i = i + 1
endwhile
arr = temporary(arr[0:i-1])
```

This way is not always good because one have to think of maximum
number of data and allocate huge superfluous memory. The next idea is
to lengthen array by the lump only on demand. I am using for this
purpose my simple "AS IS" procedure.

```
pro AddItem, arr, Q, item, step, clear = clear
  ; arr - array of data
  ; Q - number of valid elements in array
  ; item - item to be added
  ; step - size of superfluous elements of array
  ; /clear - initialize arr and Q (if they exist before)
```

```
  if keyword_set(clear)  then begin
    if n_elements(arr) NE 0 then tmp = temporary(arr)
    Q = 0L
    return
  endif
  Qitem = n_elements(item)
  Qarr = n_elements(arr)
  if Qarr EQ 0 then begin
    arr = [item,replicate(item[0],step)]                ; first
add to arr
    Q = Qitem
  endif else begin
    if Qarr GE (Q+Qitem) then $
      arr[Q:Q+Qitem-1] = item $                         ; item fit
in arr
    else $
      arr = [arr[0:Q-1],item,replicate(item[0],step)] ; item do not
fit in arr, extending
    Q = Q + Qitem
  endelse
end
```

AddItem usage:

```
  AddItem, arr, Qarr, /clear

  while ... do begin
    AddItem, arr, Qarr, item, 10000L
  endwhile

  arr = temporary(arr[0:Qarr-1])
```

You can rewrite AddItem as object if you will. I guess it is done already by someone.

What is interesting for me is file version of this procedure. Sometimes I need file that keeps different data arrays. I need to add, insert, and delete items from arrays in this file. And it should be relatively fast. I understand that this task is solved by database management systems but often using DBMS seems very excessive. And it would be nice to have save/restore style of parameters setting. Does anyone have written something like that?

Regards,
Altyntsev Dmitriy

Subject: Re: array concatenation and optimization
Posted by R.Bauer on Thu, 27 Sep 2001 07:25:11 GMT
View Forum Message <> Reply to Message

Sean Raffuse wrote:
>
> Hello.
>
> I am trying to read a bunch of data from a file to a structure array. I'm
> not sure many data entries the file will have until I have read it and so I
> am increasing the size of the structure array after reading each line. I do
> this by concatenating.
>
> adp_struct_single is the structure as a "scalar"
> adp_struct is the array
>
> I concatenate like so:
> adp_struct =[adp_struct, adp_struct_single]
>
> This is working but it has increased the processing time of my loop by an
> order of magnitude. Is there a better way to do this? Is there a reason
> this is so slow?
>
> Thanks in advance.
>
> -Sean Raffuse

Dear Sean,

if you are able to count in str by e.g. a special sign
how many times you have to repeat the structure then
I believe you can read with reads the whole file
at once.

A similiar mehtod I am using in my read_data_file for ASCII data.

```
pro read_my_data,filename


  OPENR, lun, filename[0], /GET_LUN,error=err
  IF err  NE 0 THEN GOTO, help_open
  stats = FSTAT(lun)


  data=make_array(stats.size,/byte)
  read_u,lun,data
```

FREE_LUN, lun

str=bytes2strarr(data)

reads,str,structure

end

This routines from our library may be interesting you.

http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_ html/dbase/download/bytes2strarr.tar.gz
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_
html/dbase/download/read_data_file.tar.gz

regards

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-1)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg1/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

http://www.fz-juelich.de/zb/text/publikation/juel3786.html
========== ======

read something about linux / windows
http://www.suse.de/de/news/hotnews/MS.html

Subject: Re: array concatenation and optimization
Posted by Craig Markwardt on Thu, 27 Sep 2001 16:34:52 GMT
View Forum Message <> Reply to Message

Paul van Delst <paul.vandelst@noaa.gov> writes:
>> This is working but it has increased the processing time of my loop by an
>> order of magnitude. Is there a better way to do this? Is there a reason
>> this is so slow?

>
> A while back someone else in this newsgroup more knowledgable than
> me (I think Craig?) explained why this is a bad method when the
> number of points is very large. Suffice it to say for something like
> up to maybe a couple hundred points, concatenation is o.k. Any more
> and you'd be needlessly sucking up CPU cycles shifting/concatenating
> large blocks of data on the fly.

Yes, this was me, among a lot of other people on the newsgroup. I
have always been a fan of "chunking," which means to read your data in
reasonably large chunks. Instead of reading 1 line at a time, read
1000 lines at a time.

The whole point is that you want your loop overhead to be smaller than
the actual work you do in the loop. By having an "x = [x, new]"
command at every step in the loop, most of the time of the loop is
spent allocating, appending, assigning, and unallocating.

My pet favorite is to read the file line by line, but grow the array
in chunks. I usually grow it by powers of two until a certain limit.
Example (not tested),

```
nmax = 100L             ;; Start array with 100 elements
xarray = fltarr(nmax)
ntot = 0L
while NOT eof(unit) do begin
  readf, unit, x
  if nmax LE ntot then begin
    ;; Grow the array if needed
    nnew = nmax < 500000 ;; Double the array, up to 500,000 elements
    xarray = [temporary(xarray), fltarr(nnew)]
    nmax = nmax + nnew
  endif
  ;; Assign the array
  xarray(ntot) = x
  ntot = ntot + 1
endwhile
```

Craig

--

 -- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------