Subject: Re: slow file-handling
Posted by David Theil on Mon, 02 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Jorn Helbert wrote:
>
> Hi,
> I have a problem reading in a large data-file
> if i just read it
> line by line it takes nearly ten minutes on a sparc twenty. my read
> block is the following
> <CODE>
> WHILE NOT eof(u1) DO BEGIN
> readf,u1,t_temp,foo,foo,foo,b_temp
> t_u = [t_u,t_temp]
> b_u = [b_u,b_temp]
> END
>
> Any ideas how to speed this up?

As long as you are doing this with a loop, this will be slow.
In general IDL is pretty slow with loops.

Option 1:
Read it in once the slow way and then save it as a binary file for
future reading, either using 'save' or 'assoc'.

Option2:
If you have many different text files you have to work with, you
might want to use a spawned or "call_external" called c or fortran
program to read them and then rewrite them as binaries.
Option3:
Another possibilty that will MIGHT be faster is to define a
structure that matches each row of data and then create an array
of this structure with one element for each line.

```
a=replicate{temp, t:double(0.0), foo1:double(0.0), foo2:double(0.0), $
 foo3:double(0.0), bx:double(0.0), by:double(0.0), bz:double(0.0), $
 nmuberofrows}

openr,u1,'text.file'
readf,u1,a
close,u1
t_u = a(*).t
b_xu = a(*).bx
b_yu = a(*).by
b_zu = a(*).bz
```

and then extract the info you really want out of the structure, "a"

You can find out the length of your files using some sort of system
command, like "wc" in UNIX. If you spawn this and write the output to
another file and read THAT to determine numberofrows you can make the
whole thing an automated program. Otherwise you will get an eof error if
numberofrows is too large, although the entire file will still be read
into "a".

option 2 is probably the quickest but also the most pain to implement.
It is worthwhile if this is a program where speed is critical and you
will have to read many different files every time you run.

David Theil

home
3360 34th St. D
Boulder, Colorado
80301
(303)449-3113

work
University of Colorado
campus box 389
Boulder, Colorado
80309
(303)492-0895

No guts, no glory.

Subject: Re: slow file-handling
Posted by Marty Ryba on Tue, 03 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

In article <322ABD58.3F93@ic.ac.uk>, Jorn Helbert <jorn@ic.ac.uk> writes:
|> line by line it takes nearly ten minutes on a sparc twenty. my read
|> block is the following
|>   b_temp = double([0.,0.,0.])
|>   t_temp = [0.0]
|>   WHILE NOT eof(u1) DO BEGIN
|>   readf,u1,t_temp,foo,foo,foo,b_temp
|>   t_u = [t_u,t_temp]
|>   b_u = [b_u,b_temp]
|>   END

   You may be getting slowed by all the reallocation. You will also
get horrendous memory fragmentation. Do this:

t_u = fltarr(50)

```
b_u = dblarr(50,3)
npts = 0L
cursize = 50L
b_temp = dblarr(3)
t_temp = 0.0
WHILE NOT eof(u1) DO BEGIN
   readf,u1,t_temp,foo,foo,foo,b_temp
   t_u(npts) = t_temp
   b_u(npts,*) = b_temp
   npts = npts + 1L
   IF npts EQ cursize THEN BEGIN
 t_u = [temporary(t_u), fltarr(50)]
 b_u = [temporary(b_u), dblarr(50,3)]
 cursize = cursize + 50L
   ENDIF
ENDWHILE

t_u = (temporary(t_u))(0:npts-1L)
b_u = (temporary(b_u))(0:npts-1L,*)
```

I generally use 50 as a block size, but you could go larger.

--

Dr. Marty Ryba | Of course nothing I say here is official
MIT Lincoln Laboratory | policy, and Laboratory affililaton is
ryba@ll.mit.edu | for identification purposes only,
| blah, blah, blah, ...

Subject: Re: slow file-handling
Posted by Marty Ryba on Tue, 03 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

In article <322ABD58.3F93@ic.ac.uk>, Jorn Helbert <jorn@ic.ac.uk> writes:
|> line by line it takes nearly ten minutes on a sparc twenty. my read
|> block is the following
|> b_temp = double([0.,0.,0.])
|> t_temp = [0.0]
|> WHILE NOT eof(u1) DO BEGIN
|> readf,u1,t_temp,foo,foo,foo,b_temp
|> t_u = [t_u,t_temp]
|> b_u = [b_u,b_temp]
|> END

You may be getting slowed by all the reallocation. You will also get horrendous memory fragmentation. Do this:

```
t_u = fltarr(50)
```

```
b_u = dblarr(50,3)
npts = 0L
cursize = 50L
b_temp = dblarr(3)
t_temp = 0.0
WHILE NOT eof(u1) DO BEGIN
   readf,u1,t_temp,foo,foo,foo,b_temp
   t_u(npts) = t_temp
   b_u(npts,*) = b_temp
   npts = npts + 1L
   IF npts EQ cursize THEN BEGIN
 t_u = [temporary(t_u), fltarr(50)]
 b_u = [temporary(b_u), dblarr(50,3)]
 cursize = cursize + 50L
   ENDIF
ENDWHILE

t_u = (temporary(t_u))(0:npts-1L)
b_u = (temporary(b_u))(0:npts-1L,*)
```

I generally use 50 as a block size, but you could go larger.

--

Dr. Marty Ryba | Of course nothing I say here is official
MIT Lincoln Laboratory | policy, and Laboratory affililaton is
ryba@ll.mit.edu | for identification purposes only,
| blah, blah, blah, ...

Subject: Re: slow file-handling
Posted by jlaw on Tue, 03 Sep 1996 07:00:00 GMT

View Forum Message <> Reply to Message

In article 1AA7@star.sr.bham.ac.uk, James Tappin <sjt@star.sr.bham.ac.uk> writes:

> Jorn Helbert wrote:

```
>>     b_temp = double([0.,0.,0.])
>>     t_temp = [0.0]
>>     WHILE NOT eof(u1) DO BEGIN
>>        readf,u1,t_temp,foo,foo,foo,b_temp
>>        t_u = [t_u,t_temp]
>>        b_u = [b_u,b_temp]
>>     END
>>  Any ideas how to speed this up?
```

> One possibility on a unix box is to spawn "wc" to see how many lines there are
> (if you need VMS portability, then select that or FSTAT according to the
> operating system [!version.os] -- I don't know about PC's & macs)

I use:-
OPENR,UNIT,filename
TEMP=''
COUNT=0
WHILE NOT EOF(UNIT)
READF,UNIT,TEMP
COUNT=COUNT+1
ENDWHILE
CLOSE,UNIT

The time is probably spent in re-allocating your t_u,b_u arrays,
not in the reading.

Suck it and see...

- ;-} --- :-{ --- ;-} --- :-{ --- ;-} -
Warning! Your Operating System is out of date!
It has been superseded by a more memory intensive Operating System.
Please contact your System Vendor for details.
--- ;-} --- :-{ --- ;-} --- :-{ --- ;-}

Subject: Re: slow file-handling
Posted by Hermann Mannstein on Tue, 03 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Jorn Helbert wrote:
>
> Hi,
> I have a problem reading in a large data-file
> the file consist of 57422 rows with the following format
> t foo foo foo bx by bz
>
> all double floating and I need the foo-values not. if i just read it
> line by line it takes nearly ten minutes on a sparc twenty. my read
> block is the following
> b_temp = double([0.,0.,0.])
> t_temp = [0.0]
> WHILE NOT eof(u1) DO BEGIN
> readf,u1,t_temp,foo,foo,foo,b_temp
> t_u = [t_u,t_temp]
> b_u = [b_u,b_temp]
> END
>
> I don't know before how long the file is. so I have to use the temporary
> values.
> Any ideas how to speed this up?

How about

```
fs=fstat(u1)
n_lines=fs.size/(7*4)  ; size / bytes per line - if there are extra
    ; bytes you have to change this number
line={b:b_temp,foo:b_temp,t:t_temp}
all = replicate(line,n_lines_
readf,u1,all
free_lun,u1
```

should be faster, and you can acces your fields from the structure (a.e. all(i).b)

--
Regards,

~~~~~~
Hermann Mannstein   Tel.: +49 8153 28-2503
Institut fuer Physik der Atmosphaere   or -2558
DLR - Oberpfaffenhofen   Fax.: +49 8153 28-1841
Postfach 1116   \   mailto:H.Mannstein@dlr.de
D-82230 Wessling   \ 0   http://www.op.dlr.de/~pa64
Germany   ________\/|________
~~~~~~`--------\--------'~ ~~~~~~~~~~~~~~~~~~~~
\

Subject: Re: slow file-handling
Posted by James Tappin on Tue, 03 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Jorn Helbert wrote:

```
>
> Hi,
> I have a problem reading in a large data-file
> the file consist of 57422 rows with the following format
> t foo foo foo bx by bz
>
> all double floating and I need the foo-values not. if i just read it
> line by line it takes nearly ten minutes on a sparc twenty. my read
> block is the following
>    b_temp = double([0.,0.,0.])
>    t_temp = [0.0]
>    WHILE NOT eof(u1) DO BEGIN
>       readf,u1,t_temp,foo,foo,foo,b_temp
>       t_u = [t_u,t_temp]
>       b_u = [b_u,b_temp]
```

>   END
>
> I don't know before how long the file is. so I have to use the temporary
> values.
> Any ideas how to speed this up?
>
> cheers
> jorn
>
> --
> Joern Helbert
>
> Space and atmospheric physics group
> Imperial College - London
> phone: 0171 594-7764

One possibility on a unix box is to spawn "wc" to see how many lines there are
(if you need VMS portability, then select that or FSTAT according to the
operating system [!version.os] -- I don't know about PC's & macs)

So you'd have something like:

```
spawn, 'wc '+filename, res
nrecs=fix(res)       ; Number of lines is first field in WC output

inarr =dblarr(7,nrecs)

openr, iu, /get, filename
readf, iu, inarr

t_u = transpose(float(inarr(0,*)))   ; Assuming you really need to go
        ; back to single precision, transpose
        ; Otherwise it will be a (1,n) array
b_u = inarr(4:6, *)
```

--
 +-----------------------+---------------------------------- --+---------+
James Tappin,	School of Physics & Space Research	O__
sjt@star.sr.bham.ac.uk	University of Birmingham	-- \/`
Ph: 0121-414-6462. Fax: 0121-414-3722	
 +-- --+---------+

Subject: Re: slow file-handling
Posted by brian.jackel on Wed, 04 Sep 1996 07:00:00 GMT

> In article <322ABD58.3F93@ic.ac.uk>, Jorn Helbert <jorn@ic.ac.uk> writes:
> |> line by line it takes nearly ten minutes on a sparc twenty. my read
> |> block is the following
> |> b_temp = double([0.,0.,0.])
> |> t_temp = [0.0]
> |> WHILE NOT eof(u1) DO BEGIN
> |> readf,u1,t_temp,foo,foo,foo,b_temp
> |> t_u = [t_u,t_temp]
> |> b_u = [b_u,b_temp]
> |> END

Using FSTAT() to determine the number of lines works just fine on
a PC, your mileage may vary on other platforms. For example

```
IDL>  OPENR,1,file1
IDL> help,/str,fstat(1)
** Structure FSTAT, 12 tags, length=36:
   UNIT            LONG                 1
   NAME            STRING    'f:\midas\aug23.96\millstone\file12.pwr'
   OPEN            BYTE         1
   ISATTY          BYTE         0
   ISAGUI          BYTE         0
   INTERACTIVE     BYTE         0
   READ            BYTE         1
   WRITE           BYTE         0
   TRANSFER_COUNT  LONG                 0
   CUR_PTR         LONG                 0
   SIZE            LONG           2704156
   REC_LEN         LONG                 0
```

Which gives the filesize, and the fact we're at the beginning.
Then make something to read in each line, in my case it's
just an array, you may need a structure of some kind.

```
IDL>  line= FLTARR(710)
IDL>    READF,1,line
```

After that, the filestatus shows:

```
IDL> help,/str,fstat(1)
** Structure FSTAT, 12 tags, length=36:
   UNIT            LONG                 1
   NAME            STRING    'f:\midas\aug23.96\millstone\file12.pwr'
   OPEN            BYTE         1
   ISATTY          BYTE         0
   ISAGUI          BYTE         0
   INTERACTIVE     BYTE         0
```

```
READ          BYTE        1
WRITE         BYTE        0
TRANSFER_COUNT  LONG            710
CUR_PTR       LONG          11362
SIZE          LONG        2704156
REC_LEN       LONG              0
```

Where according to the help file:

TRANSFER_COUNT is
The number of scalar IDL data items transferred in the last input/output operation on the unit. This is set by the following IDL routines: READU, WRITEU, PRINT, PRINTF, READ, and READF. TRANSFER_COUNT is useful when attempting to recover from input/output errors.

and CUR_PTR tells you how many bytes were read in total. So long as there's no extra stuff at the end of the file, just divide REC_LEN by CUR_PTR to get the total number of lines. After that, allocate a large enough block, and read in everything at once.

Please let me know if this doesn't work, I'd been assuming I could move this to other platforms, and would like to know if I'm in for trouble :)

Brian Jackel
