Subject: Re: BIL to BSQ in chunks
Posted by penteado on Fri, 04 Sep 2009 01:50:52 GMT
View Forum Message <> Reply to Message

On Sep 3, 2:32 pm, RATS <rafal...@gmail.com> wrote:
> Hi all,
>
> I am trying to convert very large BIL files to BSQ. The files are too
> big to open in one single shot so I have to block them.
> Each block is then converted to BSQ.
>
> The dimensions of a file are:
> Samples: 296
> Lines: 8000
> Bands: 492
> Data type: UINT
>
> My question is: Is there a way to open a file for writing and leave it
> open while keep adding the BSQ chunks ?
> Here is what I was trying to do, but with an unsuccessful result ... :

It is not a problem to keep it open while you add stuff to it. It is
what the
code you wrote does. The problem here is that the last dimension has
changed.
The chunks you are reading are dividing the array over its last
dimension. But
when you write that way into the file, the transposed chunks in the
output are
appended over the output's last dimension.

That is, you are reading an array of the form

```
 0  1
 2  3
 4  5

 6  7
 8  9
10  11

12  13
14  15
16  17

18  19
20  21
22  23
```

Which you want to turn into

```
IDL> print,transpose(a,[0,2,1])
  0   1
  6   7
 12  13
 18  19

  2   3
  8   9
 14  15
 20  21

  4   5
 10  11
 16  17
 22  23
```

So the problem is that to write the first chunk of the ouput:

```
  0   1
  6   7
 12  13
 18  19
```

You need to read non-consecutive parts of the input. Which means that each
output chunk requires every 3rd row from the input - 3 because that is the
2nd dimension of the input, that is to become the 3rd dimension in output. In
your case, each chunk should be made of rows read with a step of
492 lines on the input. Then the input file has to be rewinded after reading
each chunk:

```
ns=296
nl=8000
nb=492
arr=uintarr(ns,nb)
openr,lun,file,/get_lun
openw,out,'OUTPUT_FILE',/get_lun
for j=0,nb-1 do begin ;each pass in this loop writes nsXnl elements
  point_lun,lun,0
  for i=0,nl-1 do begin
    readu,lun,arr ;read nsXnb elements
    writeu,out,arr[*,j] ;write only the proper ns elements
```

```
  endfor
endfor
free_lun,lun
free_lun,out
```

This is just to give the idea of how the order of the elements read relates to the order they are written. Actually writing it like that would be horribly inefficient: this only keeps 296x492 elements in memory at a time, and reads the entire file 492 times. You need to make the number of elements read at a time as large as you can fit in memory, to decrease the number of passes through the input file.

Subject: Re: BIL to BSQ in chunks
Posted by penteado on Fri, 04 Sep 2009 03:38:54 GMT
View Forum Message <> Reply to Message

On Sep 3, 10:50 pm, pp <pp.pente...@gmail.com> wrote:
> This is just to give the idea of how the order of the elements read
> relates to the order they are written. Actually writing it like that
> would be horribly inefficient: this only keeps 296x492 elements in
> memory at a time, and reads the entire file 492 times. You need to
> make the number of elements read at a time as large as you can fit in
> memory, to decrease the number of passes through the input file.

Now, this is a decent way to do it, though it is far less obvious how it works:

```
ns=296L ;samples
nb=492L ;bands
nl=8000L ;lines
nc=2L ;chunk size
ncs=nb/nc ;number of chunks
len=2 ;size in bytes of one element (2 bytes for uint)
c=uintarr(ns,nl,nc)
d=uintarr(ns*nc)
openr,lun,file,/get_lun
openw,out,'OUTPUT_FILE',/get_lun
inds=lindgen(ns)
for j=0L,ncs-1 do begin
  for i=0L,nl-1 do begin
    point_lun,lun,((j*ns*nc)+i*ns*nb)*len
    readu,lun,d
    for k=0L,nc-1 do c[ns*(k*nl+i)]=d[inds+k*ns]
  endfor
  writeu,out,c
endfor
free_lun,lun
```

free_lun,out

The chunk size (nc) determines how much memory is used (ns*nl*nc), and how many times the input file is rewinded (ncs, the number of bands divided by the chunk size, which must be an integer). That is, a chunk is the number of indexes in the 3rd dimension of the output that are spanned on each pass through the input file.

This is efficient because it does not read anything that is not used (the jumps in input are made pointing to the next place to read from), and it rewinds the input file only ncs times.

For instance, if half the array fits in memory, the chunk size should be half the number of bands, and the program passes through the input file twice.

Now, a question: Is there a function that returns the size in bytes of a function that returns the number of bytes each IDL type uses? I do not know of any, so I typed in the 2 bytes that uint takes, for this case. Though it is easy to write one that just looks up the size for all the constant-sized types.

Subject: Re: BIL to BSQ in chunks
Posted by rafaloos on Fri, 04 Sep 2009 15:19:44 GMT
View Forum Message <> Reply to Message

Thank you for all your help.
I thought I could avoid all the loops to speed up the process ... but your code does the job really fast.
It took less then 4 minutes to covert 8089 lines. :)

Subject: Re: BIL to BSQ in chunks
Posted by penteado on Fri, 04 Sep 2009 16:58:28 GMT
View Forum Message <> Reply to Message

On Sep 4, 12:19 pm, RATS <rafal...@gmail.com> wrote:
> Thank you for all your help.
> I thought I could avoid all the loops to speed up the process ... but
> your code does the job really fast.
> It took less then 4 minutes to covert 8089 lines. :)

I was just noticing that with those dimensions you files are ~2.2GB.
So the variables I used to calculate the file location (for point_lun) are getting close to overflow. It would be better to change the dimensions and indexes in the code I sent to type 64-bit integer:

ns=296LL ;samples
nb=492LL ;bands
nl=8000LL ;lines
nc=2L ;chunk size

That way their products will not overflow if the file is a little
larger.

Subject: Re: BIL to BSQ in chunks
Posted by penteado on Fri, 04 Sep 2009 17:03:03 GMT
View Forum Message <> Reply to Message

On Sep 4, 1:58 pm, pp <pp.pente...@gmail.com> wrote:
> That way their products will not overflow if the file is a little
> larger.

Actually I should have said that you need to make that change now, if
you have not already, because with the dimensions you gave the
argument to point_lun is overflowing at the end (it is a bit more than
2^31-1).

Subject: Re: BIL to BSQ in chunks
Posted by rafaloos on Fri, 04 Sep 2009 17:22:16 GMT
View Forum Message <> Reply to Message

On Sep 4, 10:03 am, pp <pp.pente...@gmail.com> wrote:
> On Sep 4, 1:58 pm, pp <pp.pente...@gmail.com> wrote:
>
>> That way their products will not overflow if the file is a little
>> larger.
>
> Actually I should have said that you need to make that change now, if
> you have not already, because with the dimensions you gave the
> argument to point_lun is overflowing at the end (it is a bit more than
> 2^31-1).

Yes ... I made those changes ...
I also added the check when calculating the first number of loops. In
my case, the chunk is 100 so I added this line: "rest = bands mod
chunk". Like this I can go through the last 92 bands in my case.

Thank you very much again :)
