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.
