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.