EGU22-13259

Adding Quantization to the NetCDF C and Fortran Libraries to Enable Lossy Compression

Edward Hartnett[1,2], Charlie Zender[3]

1 CIRES, University of Colorado, Boulder, Colorado 80309, USA
2 NOAA/NWS/NCEP/EMC,College Park, Maryland 20740, USA
3 Departments of Earth System Science and Computer Science, University of California, Irvine, Irvine, CA 92697-3100, USA

Lossy Compression with Quantization

Quantization + Compression = Lossy compression

Why Quantize?

- By setting unused bits to constant value we make compression work better.
- This changes the data enabling lossy compression.
- Support for bitgrooming has been added to C and Fortran netCDF libraries.

32 and 64 bit floating points can carry 7/14 significant digits – more than is generally required for science needs. When compressing the data, we spend much time compressing bits we don't care about.

Notes on Quantization

- Works for netCDF/HDF5 files (i.e. files created with NC_NETCDF4 flag).
- Quantize functions are currently working in netcdf-c main branch, and will be released with the next version of the C library (4.9.0).
- Once released in netcdf-c, the Fortran quantize functions will be in the main branch of netcdf-fortran, and will be released with the next netcdf-fortran release.
- Three quantization algorithms are available in netcdf-c.
- Quantization (like compression) must be turned on after a variable is defined, before enddef is called (i.e. before any data are written to the variable.

More Notes on Quantization

- Only for NC_FLOAT, NC_DOUBLE types.
- If a fill value is set with _FillValue attribute, then any value that matches the fill value will not be quantized.
- If _FillValue is not present, then default fill values are used, and will not be quantized.
- Quantize works and is tested with parallel I/O.
- Currently works with netCDF-4/HDF5 data. Planned for Zarr in a future release.
- Turning on quantize causes an attribute to be added.
- Quantized data are fully backward-compatible, and can be read correctly by all versions of netCDF and netCDF-Java.

Turn on Quantize for a Variable

nsd Number of significant digits to retain. Allowed single- and double-precision NSDs are 1-7 and 1-15, for BITGROOM/GRANULARBR, and 1-23 and 1-52 for BITROUND.

Quantize Algorithms

- NC_QUANTIZE_BITGROOM
- NC_QUANTIZE_BITROUND
- NC_QUANTIZE_GRANULARBR

```
int nc_def_var_quantize(int ncid,
    int varid, int quantize_mode, int nsd);
```

Quantize Attributes

Turning on quantize for a variable adds an attribute, which contains the number of significant digits retained.

"_QuantizeBitGroomNumberOfSignificantDigits"
"_QuantizeGranularBitRoundNumberOfSignificantDigits"
"_QuantizeBitRoundNumberOfSignificantBits"

Inquire about Quantization

This function checks for the attribute added when quantize is used.

Note

The quantize attribute is the only way to be sure that quantization has been used with a variable.

```
int nc_inq_var_quantize(int ncid, int varid,
int *quantize_modep, int *nsdp)
```

Quantize Example - C

```
/* Create two variables, one float, one double. Quantization
 * may only be applied to floating point data. */
if (nc_def_var(ncid, "var1", NC_FLOAT, NDIM1, &dimid, &varid1)) ERR;
if (nc_def_var(ncid, "var2", NC_DOUBLE, NDIM1, &dimid, &varid2)) ERR;

/* Set up quantization. This will not make the data any
 * smaller, unless compression is also turned on. In this
 * case, we will set 3 significant digits. */
if (nc_def_var_quantize(ncid, varid1, NC_QUANTIZE_BITGROOM, NSD_3)) ERR;
if (nc_def_var_quantize(ncid, varid2, NC_QUANTIZE_BITGROOM, NSD_3)) ERR;

/* Set up zlib compression. This will work better because the
 * data are quantized, yielding a smaller output file. We will
 * set compression level to 1, which is usually the best
 * choice. */
if (nc_def_var_deflate(ncid, varid1, 0, 1, 1)) ERR;
if (nc_def_var_deflate(ncid, varid2, 0, 1, 1)) ERR;
```

Quantize Example - F77

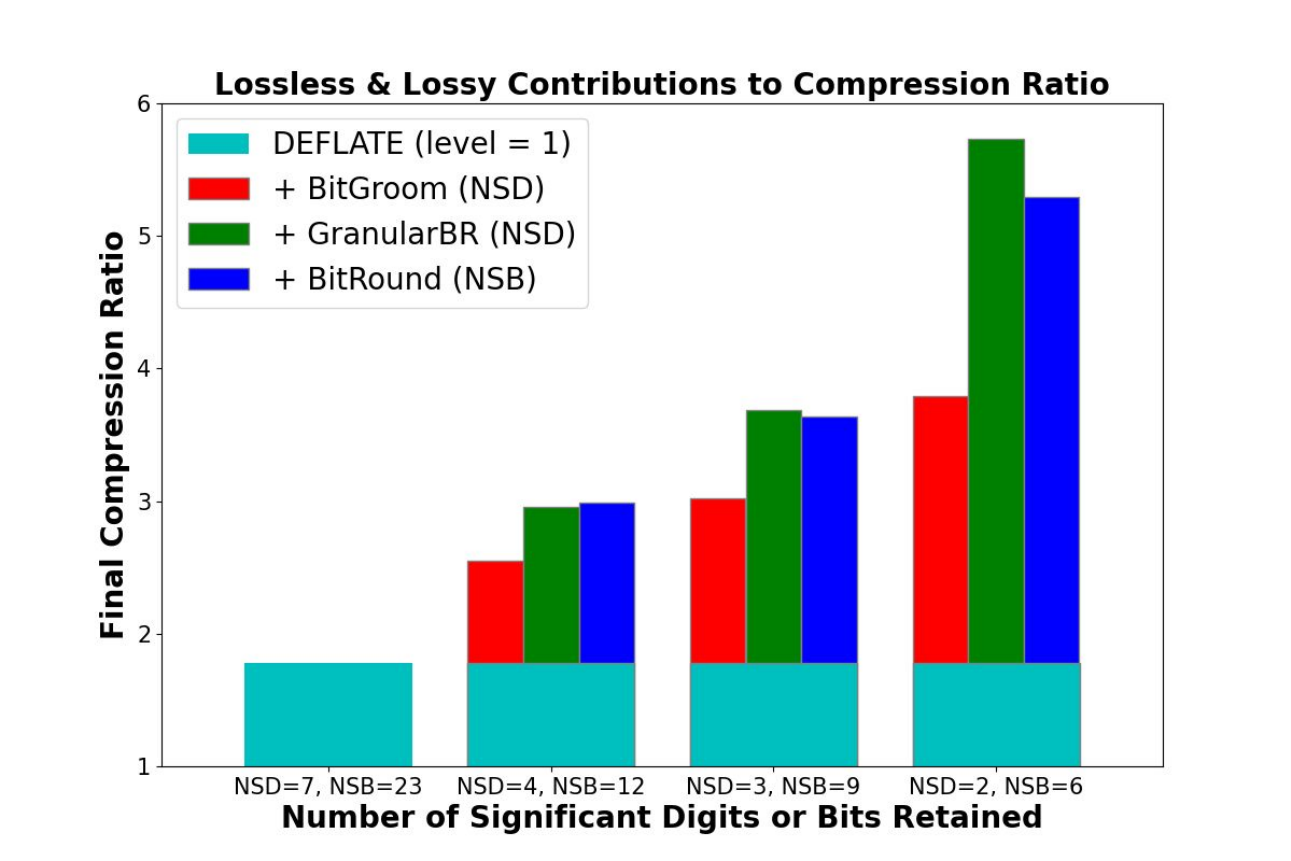
```
C     Create some variables.
      do x = 1, NVARS
          retval = nf_def_var(ncid, var_name(x), var_type(x), NDIM1,
     $         dimids, varid(x))
          if (retval .ne. nf_noerr) stop 3

C         Turn on quantize.
          retval = nf_def_var_quantize(ncid, varid(x),
     $         NF_QUANTIZE_BITGROOM, NSD_3)
          if (retval .ne. nf_noerr) stop 3

C         Turn on zlib compression.
          retval = nf_def_var_deflate(ncid, varid(x), 0, 1, 1)
          if (retval .ne. nf_noerr) stop 3
      end do
```

Quantize Example - F90

```
! Define some variables.
call check(nf90_def_var(ncid, VAR1_NAME, NF90_FLOAT, dimids, varid1&
     &, deflate_level = DEFLATE_LEVEL, quantize_mode =&
     & nf90_quantize_bitgroom, nsd = 3))
call check(nf90_def_var(ncid, VAR2_NAME, NF90_DOUBLE, dimids,&
     & varid2, contiguous = .TRUE., quantize_mode =&
     & nf90_quantize_bitgroom, nsd = 3))
```



Compression ratio of E3SM Atmosphere Model (EAM) v2 default monthly dataset of raw size 445 MB compressed with default netCDF lossless compression algorithm (DEFLATE, compression level=1) alone (leftmost), or after pre-filtering with one of three lossy codecs (BitGroom, Granular BitGroom, or BitRound) with quantization increasing (and precision decreasing) to the right.

Thanks

Extended abstract with further detail:
https://tinyurl.com/2cx66asy