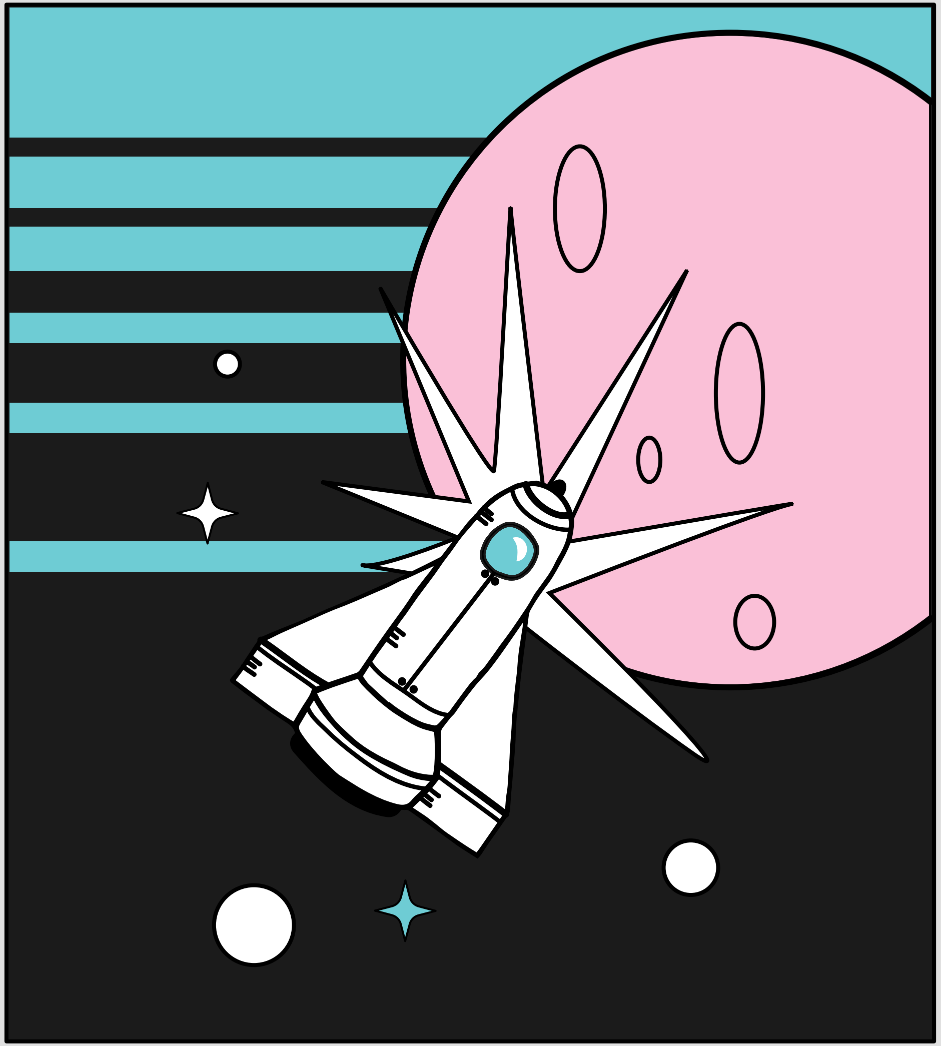
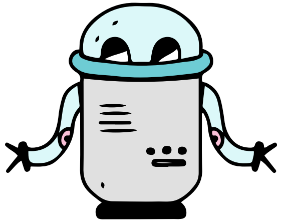
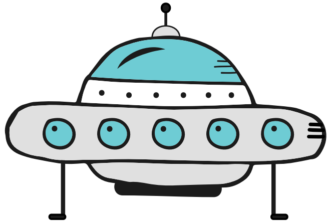
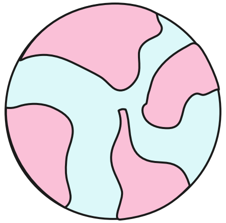
HDF5 Users Workshop 2021: Additional Compression with NetCDF

Edward Hartnett, CIRES/NOAA NCEP
Collaborators:
UCI - Charles S. Zender
UCAR - Ward Fisher, Dennis Heimbigner
NCEP - Hang Lei, Brian Curtis, Kyle Gerheiser



NetCDF Backend Formats

		
Classic Formats	**HDF5**	**Zarr**
CDF1,2,and 5. Straightforward binary file; Only classic data model supported. No compression available within netCDF.	Complex format, no size limitations, offers enhanced data model, supports zlib and szip compression.	Recently added support for the S3 cloud. Complete netCDF compatibility. Supports compression.

NetCDF Compression

NetCDF

C vs. Fortran:

- Fortran APIs are layers on top of C.
- C functionality can be enabled in the Fortran libraries easily or with no changes.
- netcdf-java is a rewrite.

Compression:

- Zlib-based compression supported since netCDF-4.0.
- szip-based compression read supported since 4.0, write since 4.6.x(?).

CCR

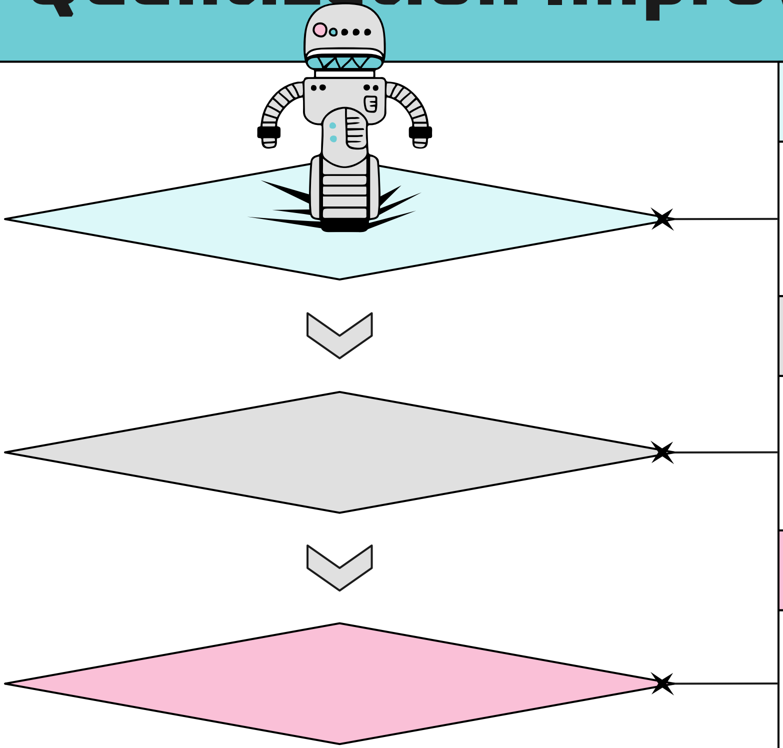
Community Codec Repository:

- Takes advantage of HDF5 filters, supported in netcdf-c since 4.8.0.
- A place for community-developed filters, including existing compression filters.

Quantization:

- Contains current big-grooming filter.
- Contains ZStandard compression.

Quantization Improves Compression



Floating Point Arrays

The 32-bit float and 64-bit double can represent up to 7/14 digits of precision. More than commonly needed.

Quantize

Define required precision and set excess bits to 0s or 1s.

Compress

Due to the longer runs of 1s and 0s in the quantized data, compression will be more effective, with smaller output.

UFS

UNIFIED FORECAST SYSTEM

36 GB

to

6 GB

UFS atmospheric model output file compresses from 36 GB to 6 GB with a combination of lossy quantization and lossless zlib compression.

Quantize with the BitGroom Filter in CCR

The bitgroom filter is currently implemented in CCR.

- Well-tested code.
- Some algorithm upgrades may be coming, will not change API.
- As a filter, only applies to netCDF-4/HDF5 files.
- Because a filter, must be installed on reading system, though it has nothing to do on read.
- Introduces incompatibilities with netcdf-java, even for read-only access.

Quantize in netcdf-c

By moving the bitgroom code to the netcdf-c library we achieve much better results.

- Used by netCDF-4/HDF5, also available to other backend formats (classic, nczarr).
- Classic formats will benefit when bitgroom is applied and then file is compressed.
- nczarr benefits for whatever compression it uses.
- Bitgroomed data immediately readable by all existing versions of netcdf-c and netcdf-java.
- Currently working in main branch of netcdf-c, netcdf-fortran changes are pending.

Turn on Quantize for a Variable

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

- Like nc_def_var_deflate(), must be defined before enddef, can't be changed after that.
- Only applies to float or double, returns error otherwise.
- Quantize modes: NC_NOQUANTIZE (0), NC_QUANTIZE_BITGROOM (1).

```
* @param nsd Number of significant digits to retain. Allowed single- and
* double-precision NSDs are 1-7 and 1-15, respectively.
```

Inquire About Quantize

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

- Works like other netCDF inq functions.

Implementation in NetCDF-4

Quantization is achieved in this function from libsrc4/nc4var.c. It copies the data element by element.

```
int
nc4_convert_type(const void *src, void *dest, const nc_type src_type,
                 const nc_type dest_type, const size_t len, int *range_error,
                 const void *fill_value, int strict_nc3)
```

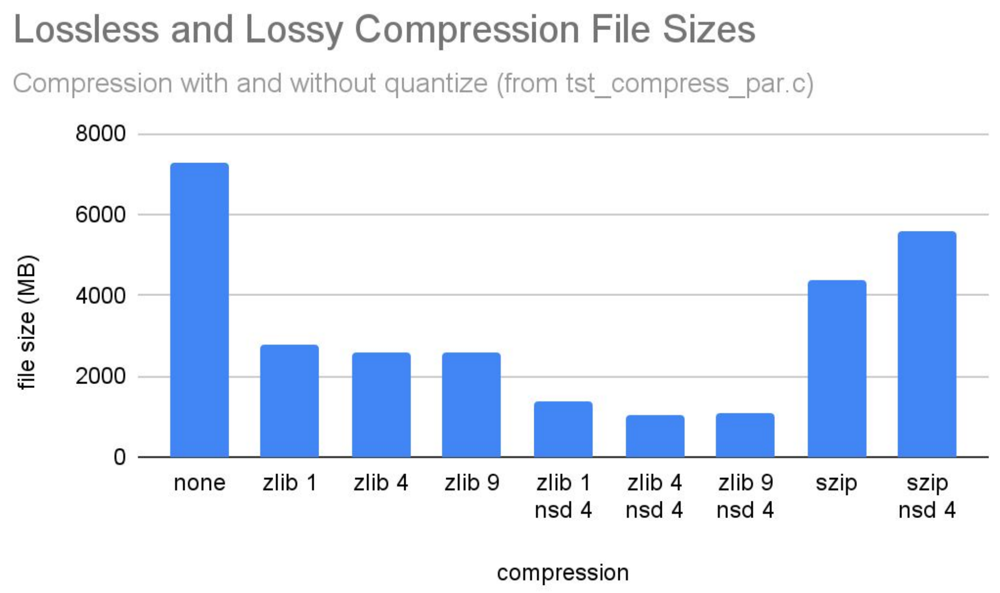
Example

```
/* Create a netcdf-4 file with two vars. */
if (nc_create(FILE_NAME, NC_NETCDF4|NC_CLOBBER, &ncid)) ERR;
if (nc_def_dim(ncid, DIM_NAME_1, DIM_LEN_5, &dimid)) ERR;
if (nc_def_var(ncid, VAR_NAME_1, NC_FLOAT, NDIM1, &dimid, &varid1)) ERR;
if (nc_def_var(ncid, VAR_NAME_2, NC_DOUBLE, NDIM1, &dimid, &varid2)) ERR;

/* Turn on quantize for both vars. */
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;

/* Write some data. */
if (nc_put_var_float(ncid, varid1, float_data)) ERR;
if (nc_put_var_double(ncid, varid2, double_data)) ERR;

/* Close the file. */
if (nc_close(ncid)) ERR;
```

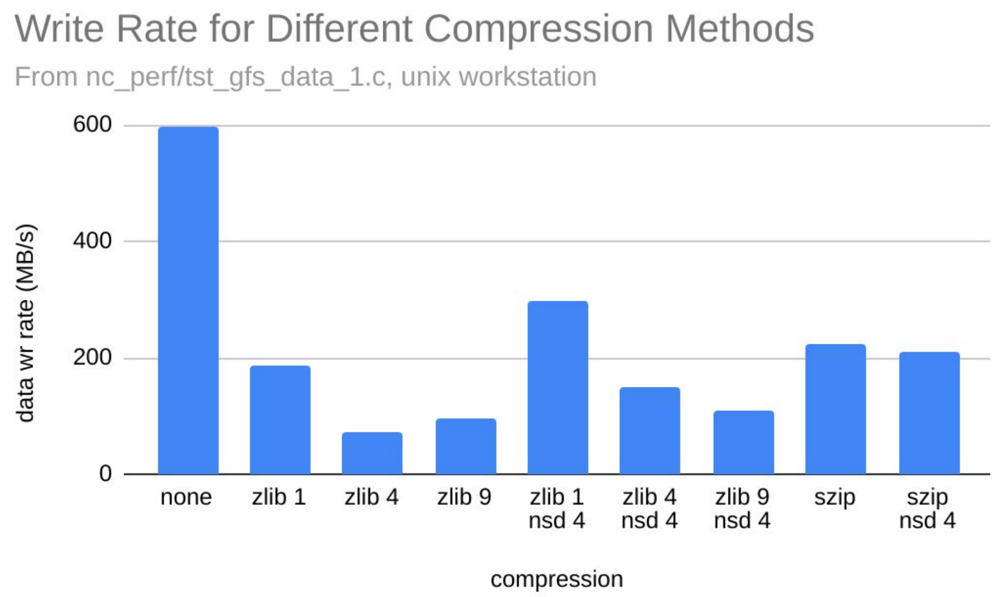
Quantize Details

- 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.
- Quantize works and is tested with parallel I/O.
- Currently works with netCDF-4/HDF5 data. Planned for Zarr in a future release.
- More efficient and effective quantization schemes may be added in future releases.
- Turning on quantize causes an attribute to be added: _QuantizeBitgroomNumberOfSignificantDigits.

Quantize Improves Compression



Quantize Improves Performance


Write Rate for Different Compression Methods
From nc_perf/tst_gfs_data_1.c, unix workstation
data wr rate (MB/s)
none
zlib 1
zlib 4
zlib 9
zlib 1 nsd 4
zlib 4 nsd 4
zlib 9 nsd 4
szip
szip nsd 4
compression