Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

Geoscientific Model Development Discussions

Finding the Goldilocks zone: Compression-error trade-off for large gridded datasets

Jeremy D. Silver[1] and Charles S. Zender[2]

[1]School of Earth Sciences, University of Melbourne, Australia
[2]Departments of Earth System Science and of Computer Science, University of California, Irvine, USA

Correspondence to: J. D. Silver (jeremy.silver@unimelb.edu.au)

Abstract. The netCDF-4 format is widely used for large gridded scientific datasets, and includes several compression methods: lossy linear scaling and non-lossy deflate and shuffle algorithms. Many multidimensional datasets exhibit considerable variation over one or several spatial dimensions (e.g. vertically) with less variation in the remaining dimensions (e.g. horizontally). On such datasets, linear scaling with a single pair of scale and offset parameters often entails considerable loss of precision. We propose a method (termed "layer packing") that simultaneously exploits lossy linear scaling and lossless compression. Layer packing stores arrays (instead of a scalar pair) of scale and offset parameters.

An implementation of this method is compared with existing compression techniques in terms of compression ratio, accuracy, and speed. Layer packing produces typical errors of 0.01-0.02% of the standard deviation within the packed layer, and yields files roughly 33% smaller than the lossless deflate algorithm. This was similar to storing between 3 and 4 significant figures per datum. In the six test datasets considered, layer packing demonstrated a better compression/error trade-off than storing 3-4 significant digits in half of cases and worse in the remaining cases, highlighting the need to compare lossy compression methods in individual applications. Layer packing preserves substantially more precision than scalar linear packing, whereas scalar linear packing achieves greater compression ratios. Layer-packed data files must be "unpacked" to be readily usable. These characteristics make layer-packing a competitive archive format for many geophysical datasets.

Keywords. NetCDF-4; HDF5; Lossy compression; Data storage format

1 Introduction

The volume of both computational and observational geophysical data has grown dramatically in recent decades, and this trend is likely to continue. While disk storage costs have fallen, strong constraints remain on data storage. Hence, in practice, compression of large datasets continues to be important for efficient use of storage resources.

Two important sources of large volumes of data are computational modelling and remote-sensing (principally from satellites). These data often have a "hypercube" structure, and storage formats such as HDF5 (http://www.hdfgroup.org), netCDF-4 (http://www.unidata.ucar.edu/netcdf) and GRIB (http://rda.ucar.edu/docs/formats/grib2/grib2doc/). Each of these has their own built-in compression techniques, allowing data to be stored in a compressed format, while simultaneously allowing access to the data (i.e. incorporating the compression/decompression algorithms into the format's API).

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

Geoscientific Model Development Discussions
Open Access
EGU

The HDF5 and netCDF-4 formats provide a highly flexible framework, allowing for attributes, groups, and user-defined data types. Variables can be defined with as many as 32 or 1024 dimensions (respectively for the two formats); the relatively free framework to define meta-data allows such formats to be self-documenting and applicable for many contexts. NetCDF-4 is built upon HDF5, and inherits most of its capabilities. Both HDF5 and netCDF-4 include the "deflate" compression algorithm (*Deutsch*, 2008; *Ziv and Lempel*, 1977, 1978) and a shuffle filter. These are "lossless" compression techniques, in that no precision is lost. By contrast, "linear scaling" is a "lossy" compression method (implying a loss of accuracy), which rounds the data to discrete values within a defined range. A common usage of linear scaling is to represent variables as (signed or unsigned) two-byte integers, which allows for $2^{16} = 65536$ different values. If applied to a four-byte floating point (i.e. single precision) array, this use of linear scaling cuts data usage by 50%. When the deflate and shuffle filters are also applied, the savings are typically much greater.

In many applications in the geophysical sciences, a multidimensional gridded variable may range dramatically across one dimension while exhibiting a limited range of within slices of this variable. Examples include:

- variation in atmospheric density, water vapour mixing ratio with respect to height
- variation in ocean temperature, current velocity with respect to depth
- variation in atmospheric concentrations of nitrogen dioxide with respect to height and time

We will use the term "thick dimensions" to denote those dimensions that account for the majority of the variation in such variables, "thin dimensions" to denote the remaining dimensions; in the case of the first example above (assuming a global grid and a geographic coordinate system), the vertical dimension (pressure or height) is thick, and the horizontal dimensions (latitude, longitude) and time are thin. We will use the term "thin slice" to describe a slice through the hypercube for fixed values of the thick dimensions. Note that there are cases with multiple thick dimensions, such as the third example noted above.

Linear-packing with a single scale-offset pair applied to such gridded variables results in a considerable loss of accuracy across thin slices of such variables. This is because in order to cover the scale of variation spanned by the thick dimensions, few discrete values will be spanned by the individual thin slices. At least two distinct methods can achieve a better compromise between data compression and loss of accuracy. First, one can store the data at a fixed precision (i.e. a chosen number of significant digits, or NSD). This method is known as "bit-grooming" and is detailed by *Zender* (2016) and implemented in the NCO package (*Zender*, 2008). If bit-groomed data remain uncompressed in floating-point format this coarsening will not affect the file size, however the application of the deflate/shuffle algorithms will in general improve the compressibility of the coarsened data, due to quantization to a smaller number of values.

Second, one can represent the field using a lower-precision value (e.g. as two-byte integers rather than four-byte floating point numbers) and an agreed transformation between the original and compressed representations; this process is termed "packing". In the common case of representing four-byte floats as two-byte integers, there is already a saving of 50% relative to its original representation (ignoring any overhead due to storing parameters involved in the transformation). In this case,

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

a range of transformations are readily available. The simplest is to use linear transformation with a single scale-offset pair, thereby avoiding the thick/thin dimension distinction. We call this "scalar linear packing" and it is the standard method of packing in the geoscience community. Its attributes are defined in the netCDF User Guide (*Unidata*, 2016) and it has a long and wide tradition of support, including automatic interpretation in a range of netCDF readers. Another option is to store for each variable *arrays* of scale-offset pairs, with size corresponding to the thick dimensions only. We will refer to this technique as "layer packing". In this article, we compare the merits of these lossy compression methods alongside established compression techniques. We do not consider packing procedures using application-specific transformations; although these may be efficient for particular applications, they are of less general interest and if they involve storing normalization factors (i.e. parameters for use in the transformation) then their performance is likely to perform similarly to the layer packing described above.

In Section 2, we compare the performance of bit grooming, scalar linear packing, layer packing, and deflate methods. These are assessed in terms of the compression savings and loss of precision. The choice of the compression methodology depends on other factors such as read/write speeds, portability and flexibility. These considerations are discussed in Section 3.

2 Methods

This section outlines the implementation of the layer-packing and unpacking methods, the storage format (in the implementation described), the test data sets, evaluation metrics and the performance of the compression methods on the test cases considered.

2.1 Software implementation and storage format

2.1.1 Layer packing

The code to perform layer-packing described in this article was written as stand-alone command-line tools in Python (v. 2.7.6). These are freely available on https://github.com/JeremySilver/layerpack. Beyond the standard Python installation, it requires the `numpy` and `netCDF-4` modules be installed. The compression is performed as follows:

```
ncpacklayer -d thickdim1,thickdim2 -v var1,var2,var3 original.nc packed.nc
```

Other optional flags allow for increased verbosity (`-V`), over-writing existing output files (`-O`) and defining the DEFLATE compression level (`-L`).

The mandatory `-d` flag is followed by a comma-separated list of the thick dimensions. The optional `-v` flag is followed by a comma-separated list of variables to pack. The default is to pack all variables defined along any of the thick dimensions listed. In the output file (in this example `packed.nc`) each variable that is packed (e.g. `var1`) is replaced by a trio of variables containing the arrays of packed values, scale factors and offsets. In this example, these are termed `var1__short`, `var1__scale` and `var1__offset`, with data type unsigned short (i.e. two-byte) integer, floating-point and floating-point, respectively. Suppose the original definition of `var1` is (following output format for the command line utility `ncdump`):

```
float var1(thindim1, thickdim1, thickdim2, thindim2) ;
```

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

Geoscientific Model Development Discussions

then the corresponding trio will have dimensions as follows:

```
ushort var1__short(thindim1, thickdim1, thickdim2, thindim2) ;
float var1__scale(thickdim1, thickdim2) ;
float var1__offset(thickdim1, thickdim2) ;
```

In other words, the scale and offset arrays have one element per thin slice. Data remain in netCDF format in this packed format and retain all their attributes. Data can be unpacked as follows:

```
ncunpacklayer packed.nc unpacked.nc
```

The `-d` and `-v` flags are not used, since this information is contained in the trios of packed arrays.

2.1.2 Methods compared

We will compare layer packing with a number of other compression methods, which in each case use command-line tools from the NCO bundle (*Zender*, 2008). The formats compared, the commands used to achieve these and the capitalized abbreviations, are listed below.

1. UNCOMPRESSED: Uncompressed netCDF

   ```
   ncks -3 in.nc out.nc
   ```

2. DEFLATE: Deflate compression (level 4) with shuffle filter

   ```
   ncks -4 -L4 in.nc out.nc
   ```

3. NSD2, NSD3, NSD4, NSD5: Deflate compression (level 4) with shuffle filter, and bit grooming storing 2, 3, 4, or 5 significant figures (respectively). The following yields three significant digits (NSD3).

   ```
   ncks -4 -L4 -ppc $vars=3 in.nc out.nc
   ```

4. LSSCALAR: Deflate compression (level 4) with shuffle filter, scalar linear packing for each variable

   ```
   ncpdq -4 -L4 in.nc out.nc
   ```

5. LSARRAY: Deflate compression (level 4) with shuffle filter, layer packing for selected dimensions

   ```
   ncpacklayer -L4 -v $vars -d $dims in.nc out.nc
   ```

2.2 Tests

The methods are compared with two metrics. The first relates to the compression efficiency, relative to DEFLATE above (deflate compression with shuffle filter). Compression ratios are defined relative to the file size generated by DEFLATE (i.e.

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

smaller is better). The second relates to the accuracy (or, seen another way, the error) of the compressed data relative to the original data. The error of UNCOMPRESSED is zero (as it is the reference data), as is DEFLATE since it ensures lossless compression. The remaining methods cause some loss of precision.

These errors can be summarised in different ways. In order to compare results across variables with different scales and units, the errors must be normalized somehow. A single normalization factor per variable can be used, however this has the disadvantage that the results will be heavily influenced by variation across the thick dimension; for example, if a field ranges across several orders of magnitude (e.g. values of $O(10^0)$ at one end of the thick dimension and $O(10^4)$ at the other), then errors will be highly non-uniform across the thick dimension. As such, we choose to normalize errors within each thin slice of a given variable by the standard deviation or mean of the original values across that thin slice; the rationale for normalizing by the standard deviation or the mean is discussed in section 2.4 below. For each thin slice of a given variable, the standardized error is defined as the ratio of the root mean-squared error to the standard deviation (or the mean); for a given variable, the standardized error is defined as the mean standardized error averaged across all thin slices. The error for LSARRAY is computed based on the unpacked contents of the file.

2.3 Datasets

The tests described above were applied to the following datasets:

1. `ei_mnth_an_pl_15x15_90N0E90S3585E_20080901_20081201`

   ERA-Interim reanalysis data (*Dee et al.*, 2011). Dimensions: 121×240 grid-points in the horizontal, 37 vertical layers, 16 time-points. 14 variables with these four dimensions. Thick dimensions chosen to be the time and vertical level.

2. `mozart4geos5_2011-02-01_2011-03-16.nc`

   A limited area subset from global MOZART model output (*Brasseur et al.*, 1998). Dimensions: 9×10 grid-points in the horizontal, 56 vertical levels, 172 time-points. 77 variables with these four dimensions. Thick dimension chosen to be the vertical level.

3. `wrfout_d03_2013-01-24_07:00:00`

   WRF model output (*Skamarock et al.*, 2005). Dimensions: 165×140 grid-points in the horizontal, 32 vertical levels, 1 time-point. 23 variables with these four dimensions, which constitute 85% of the data stored in the file. Thick dimension chosen to be the vertical level.

4. `MERRA300.prod.assim.inst3_3d_asm_Cp.20130601.nc`

   MERRA reanalysis product (*Rienecker et al.*, 2011). Dimensions: 144×288 grid-points in the horizontal, 42 vertical levels, 8 time-points. 11 variables with these four dimensions. Thick dimension chosen to be the vertical level.

5. `dstmch90_clm.nc`

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

Output of the mineral Dust Entrainment And Deposition (DEAD) model (*Zender et al.*, 2003). Dimensions: 94×192 grid-points in the horizontal, 28 vertical levels, 1 time-point. 15 variables with these four dimensions, together accounting for 87% of the data stored in the file. Thick dimension chosen to be the vertical level.

6. `famipc5_ne30_v0.3_00003.cam.h0.1979-01-L5.nc`

CAM-SE model output (*Dennis et al.*, 2012). Dimensions: 48602 grid-points in the horizontal, 30 vertical levels, 1 time-point. 123 variables with these three dimensions, together accounting for 87% of the data stored in the file. Thick dimension chosen to be the vertical level.

2.4 Results

Figure 1 shows the compression ratios for the six test data sets. As noted above, compression ratios are defined by normalizing by the file sizes from DEFLATE (deflate and shuffle). Without compression, files are on average 2.27 times larger than the benchmark, indicating the high efficiency of the lossless deflate and shuffle filters. Relative to the DEFLATE benchmark, the lossy methods had average compression ratios of 0.51 (NSD2), 0.66 (NSD3), 0.77 (NSD4), 0.91 (NSD5), 0.35 (LSSCALAR) and 0.66 (LSARRAY). In three data sets (1, 3, 4) the NSD4 and LSARRAY (shown in cyan and gray, respectively) produce very similar file sizes (within 4% of one another). For the other three data sets LSARRAY yields file sizes between 71% and 77% of NSD4, roughly in between the NSD2 and NSD3 file-sizes for these data sets. In each case, LSSCALAR produces the greatest compression.

Figure 1 presents the time taken by each compression method, and the time for extraction by LSARRAY (shown in white). Averaged across the cases, the median standardized compression times were 0.37 (UNCOMPRESSED), 0.91 (NSD2), 1.00 (NSD3), 0.92 (NSD4), 1.04 (NSD5), 0.74 (LSSCALAR) and 0.88 (LSARRAY), with the time from DEFLATE serving as a benchmark within each dataset. The median standardized extraction time for LSARRAY was 1.39 (using the same normalization factor).

Lossy compression comes at a cost to the precision of the underlying data. Figure 2 summarizes the standardized errors. A value of 0.01 in this figure indicates that, a mean error ratio of 0.01 across thin slices of a given variable, where the error ratio is the root-mean squared error within the thin slice divided by the standard deviation (or mean) of the original values within the thin slice.

The choice of error metric will highlight different aspects of the performance of each method. When normalizing the errors by the standard deviation of the original values, the bit-grooming methods appear to perform relatively poorly for fields where the baseline value is large relative to the within-slice standard deviation (e.g. temperatures stored with units of K, concentrations of well-mixed atmospheric trace gases such as CO_2 or CH_4). In these cases, the first few significant figures may be constant across the slice and all variation is recorded in the last few significant figures. Arguably such fields should be stored with a larger number of significant digits, however for simplicity with have used the same number of significant digits for all variables.

When normalizing the errors by the mean of the original values (or in this case, by the absolute value of the mean of the original values), the layer-packing will appear to perform poorly for fields where the variation is large relative to the baseline

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

value (e.g. mass of snow, concentrations of short-lived atmospheric trace gases with few emission sources). In light of such considerations, the distributions of errors are presented for both normalization methods (figures 2 and 3).

The median standardized errors (across all variables and all datasets, normalised by the standard deviation) for the lossy methods were $2.09 \cdot 10^{-2}$ (NSD2), $2.56 \cdot 10^{-3}$ (NSD3), $1.77 \cdot 10^{-5}$ (NSD4), $2.14 \cdot 10^{-6}$ (NSD5), $2.84 \cdot 10^{-2}$ (LSSCALAR) and $9.48 \cdot 10^{-5}$ (LSARRAY). When normalised by the mean, the median standardized errors remained similar for the bit-grooming methods, and approximately doubled for the other methods (to $5.55 \cdot 10^{-2}$ for LSSCALAR and $1.80 \cdot 10^{-4}$ for LSARRAY). One clear result is that LSSCALAR caused a dramatic loss of accuracy in many cases, much more than one might expect given the nominal precision of roughly $1/65535 \approx 1.53 \cdot 10^{-5}$ times the range.

The NSD5 method was consistently the most accurate, followed by NSD4, LSARRAY, NSD3, NSD2 and LSSCALAR. The NSD3, NSD4, NSD5 and LSARRAY methods were each accurate to 0.05% or better (of the standard deviation or mean). In many applications this is much smaller than the uncertainty of the modelled or observed data. The trade-off between error and compression is illustrated in figure 3. The lossy methods show something of a continuous gradient when considering both these aspects. In terms of the errors, the LSARRAY method falls in between NSD3 and NSD4, with compression ratios between NSD2 and NSD3 for half of these datasets and close to NSD4 for the remaining files.

The data-points in figure 3 appear at first glance roughly linear on the log-log scales. Using linear regression models, we found that the intercepts were significantly different ($p \approx 10^{-5}$), but that there was no significant difference between the slopes ($p \approx 0.4$); these comparisons were based on analysis of variance (ANOVA) F-statistics. In these models one outlier data-point was excluded (LSSCALAR for dataset 4) in order to satisfy the assumptions of the linear models. The linear slope on a log-log plot is suggestive of a power-law relationship, which would be consistent with fundamental constraints on compression potential consistent with rate-distortion theory (*Berger*, 2003).

3 Discussion

This paper introduces layer-packing as a variant of the linear scaling technique for data compression, and compares the results to this and other lossless and lossy compression techniques. The idea itself is not new and forms the basis of compression within the GRIB format, in that each field is a two-dimensional array, and compression is performed on fields individually. Although the methods described are used only on netCDF data files, they apply equally to other data formats (e.g. HDF4, HDF5). These results are presented as a proof-of-concept only.

The comparison with GRIB (especially GRIB2, which compresses via JPEG-2000, based on wavelet approximation) is especially relevant. The procedure described here is more general than that used in GRIB, in that thin-slices are not restricted to being two-dimensional. The GRIB format is rather restrictive in terms of meta-data, and is nowhere near as general or self-describing as HDF5 (or derivatives such as netCDF-4). On the other hand GRIB2 is very space efficient; *Caron* (2014) estimated that GRIB2 files are on average 44% of the size of the equivalent deflate-compressed netCDF-4 files (n.b. relative errors were not reported, which limits the comparison).

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

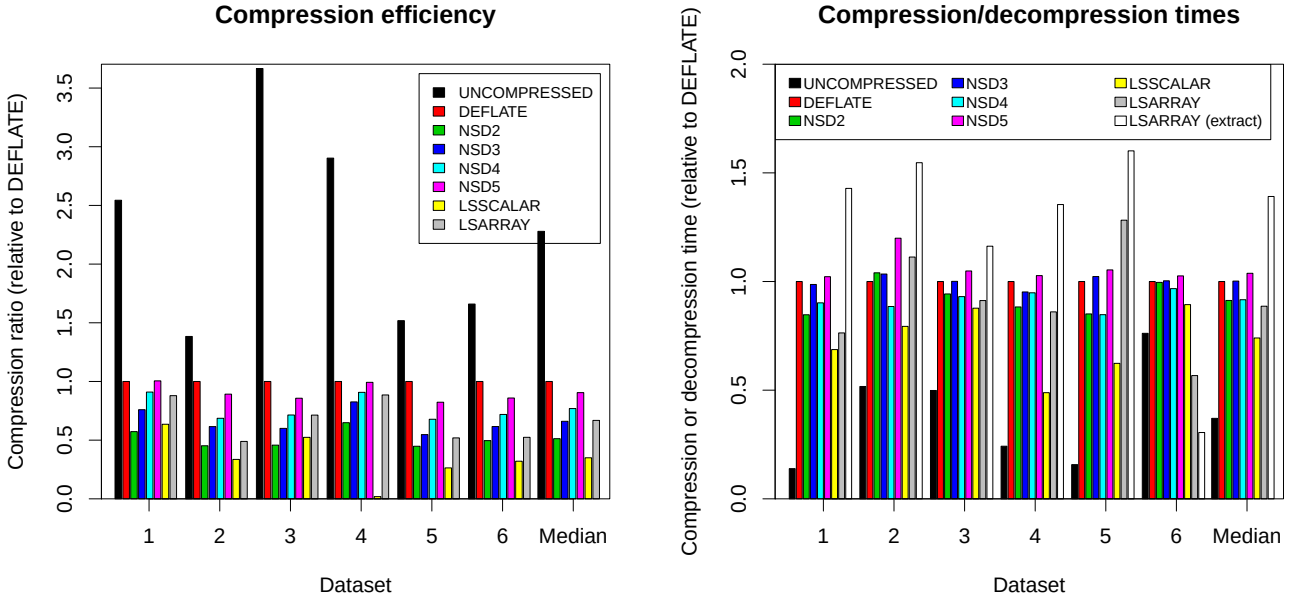

Figure 1. Left panel: Compression ratios (compressed file size divided by original file size) measured for six methods, applied to six test datasets (lower is better). The DEFLATE method serves as the benchmark and all values are equal to 1.0 by definition. Right panel: Scaled compression/decompression times for each method (lower is better), with LSARRAY represented twice (for compression and extraction). These times are normalized by the compression time from DEFLATE. The median values (shown at the right of each panel) are calculated across all datasets.

The deflate/shuffle algorithms we tested are the only two lossless filters accessible through both the HDF5 and netCDF-4 APIs, which makes their performance of the greatest interest and widest applicability. We also note that the HDF5 format allows for other filters to be used for lossless compression, not just the deflate/shuffle algorithms used here. A range of other lossless compression filters can be loaded dynamically (*Group*, 2016), many of which offer faster or more efficient compression than the default algorithms (*Hübbe and Kunkel*, 2013, e.g. as discussed by). The compression performance of the different lossy (packing or bit-grooming) filters when combined with these other lossless filters was not explored within this study but is likely to be similar to that observed for the deflate/shuffle algorithms applied here.

Both the linear scaling and the layer packing use the same representation of the data (i.e. two-byte integers), however large differences in the compression and errors were found. This is because the compressibility of a packed field is related to the distribution of values within the scaling range. This can be illustrated by the following example. Consider a three-dimensional field that ranges over three orders of magnitude in the vertical dimension, taking values of $O(10^3)$ at bottom and values of $O(10^0)$ at the top. Suppose the range across the entire field is 0.0 to 5000.0 (ignoring units), and thus the smallest increment when recorded at single precision (with linear scaling) is roughly $5000.0/65536 \approx 0.08$ across the entire field. The upper levels will be much more heavily quantized, relative to their values, compared to the lower levels, and thus will be subject to

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

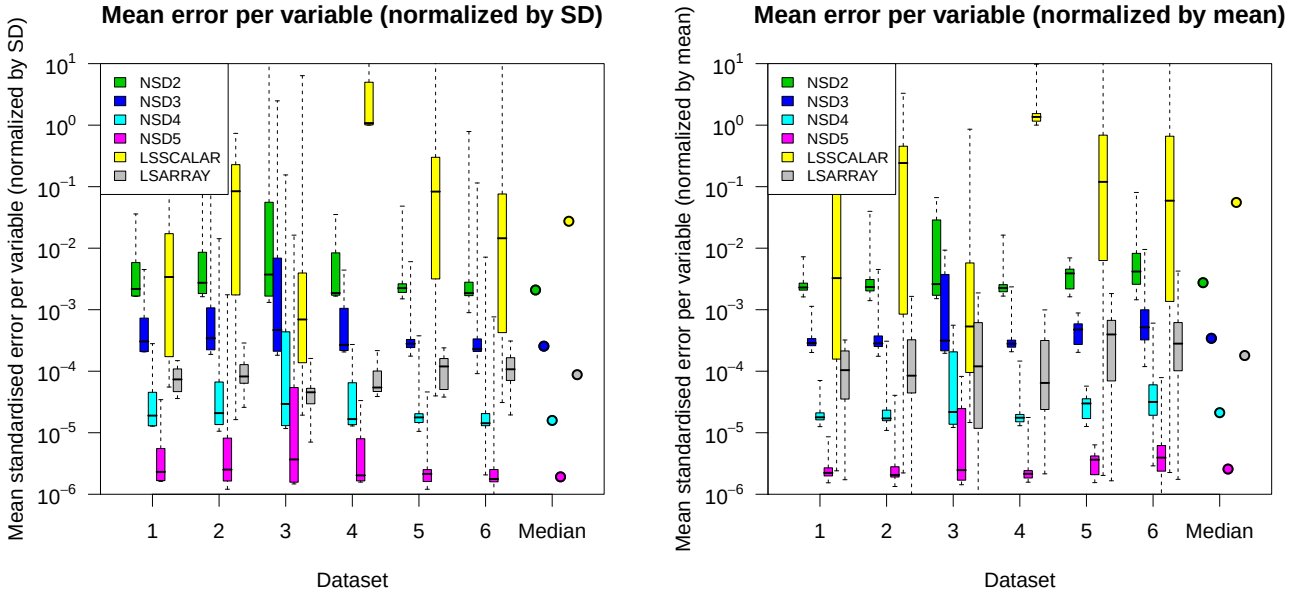

Figure 2. The distribution of standardized errors (lower is better) of the four lossy compression methods, applied to six test datasets. Each data-point behind the box-whisker plot corresponds to one variable in a given dataset, based on the mean standardized error per variable normalized by the standard deviation (left panel) or the mean absolute value (right panel) of the original data. Whiskers indicate the minimum and maximum values in the sample, box edges correspond to the 25% and 75% quantiles, and the center-line indicates the sample median. Methods are shaded with the same color as in Figure 1. The median values (shown at the right of each panel) are calculated across all variables within these datasets.

both higher compression and higher relative errors. Compressing the same field using layer packing may yield, for the sake of example, a range of 2000.0 at the bottom level and 5.0 in the top level, and in both cases the precision will be greater, and hence more values will be taken within each vertical level. As such, the more variable the field, the less compression will be obtained from the combination of the deflate and shuffle filters. Hence linear scaling gives a lower precision, less spatial variation within the thin slice and hence is more compressible. Another factor that makes layer packing less compressible is that the accompanying scale-offset parameter arrays must be stored.

Using the lossless deflate and shuffle algorithms, the compressed files were less than 50% of the original uncompressed size. Beyond this, the lossy compression methods achieved further space savings, although to varying degrees for the individual datasets. There was a general trade-off between precision-loss and compression efficiency (figure 3). When considered along both these axes, layer-packing has errors between those of NSD3 and NSD4. For half the datasets considered (1, 3, and 4) it yielded compressed sizes in between those of NSD2 and NSD3 and in the other half it yielded file sizes comparable to those of NSD4. In other words, in three of the cases considered layer-packing gave a worse compression-error trade-off (with file-size close to NSD4 but larger errors) while in remaining three cases it performed better (with both smaller errors and smaller file size than NSD3).

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

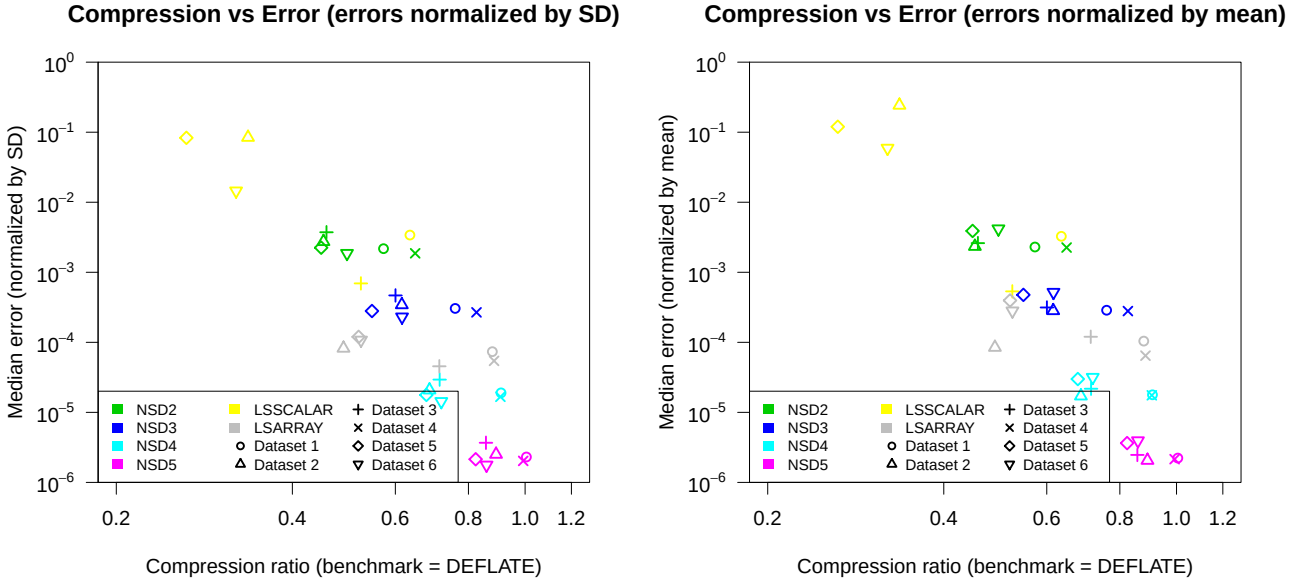

Figure 3. The median error per file (from figure 2) plotted against the corresponding compression ratio for that file (from the left panel of figure 1). Errors are normalized by the standard deviation (left panel) or the mean (right panel) of the original values within each thin slice.

Two error metrics were employed. Each emphasized different aspects of the performance. When normalizing by the mean (of the entire variable, or of the thin-slice), variables with a high standard deviation to mean ratio will show larger errors using the layer-packing compression. Whereas when normalizing by the standard deviation, variables with a high mean to standard deviation ratio will show larger errors using the bit-grooming compression. These considerations highlight the importance of understanding the properties of lossy filters when selecting the most appropriate filter for a given application.

When considering which compression method for an individual dataset, one needs to consider several factors. First, space constraints and the size of the datasets in question will vary considerably for different applications. Second, the degree of precision required will also depend on the application. The bit-grooming (storing at least three significant digits) and layer-packing techniques achieved average normalized errors of 0.05% or better, which in many geoscientific applications is much less than the model or measurement errors. Third, datasets vary considerably in their inherent compressibility. Fourth, how data are stored should also reflect how it will be used (e.g. active use versus archiving). A major disadvantage of the layer-packing as described here is that it is essentially an archive format, and needs to be unpacked by a custom application before it can be easily interpreted. Scalar linear packing is similarly dependent on unpacking, whereas bit-grooming requires no additional software. Finally, other issues relating to portability, the availability of libraries and consistency within a community also play a role in determining the most appropriate storage format.

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

4 Conclusions

This paper considers different forms of linear-scaling as a basis for compressing large gridded datasets. Layer-packing, scalar linear packing and bit-grooming were compared. Layer-packing was found to be a competitive format for archive purposes, with benefits (in terms of the compression-error trade-off) for some datasets. In most cases, it was comparable to storing between 3 and 4 significant digits per datum. Given the variation in compression and accuracy achieved for the different datasets considere, the results highlight the importance of testing compression methods on a realistic sample of the data.

If space is limited and a large dataset must be stored, then we recommend that the standard deflate and shuffle methods be applied. If this does not save enough space, then careful thought should be given to precisely which variables and which subsets of individual variables will be required in future; it often arises that despite a wealth of model output, only a limited portion will ever be examined. Many tools exist for sub-setting such datasets. Beyond this, if further savings are required and if the data need not be stored in full precision, then the bit-grooming methods should be trialled. Layer-packing should be reserved for archive applications, and should be compared with bit-grooming.

5 Code availability

The python-based command line utilities used for the layer-packing, unpacking and relative-error analysis are available freely at https://github.com/JeremySilver/layerpack.

6 Data availability

Of the datasets used in the tests (listed in section 2.3), datasets 2-6 are available online at https://figshare.com/projects/Layer_Packing_Tests/14480 along with a brief description of each file. Dataset 1 could not be distributed with the other files due to licensing restrictions but can be accessed through the ECMWF's public dataset portal (http://apps.ecmwf.int/datasets/), using the following set of inputs: stream = synoptic monthly means, vertical levels = pressure levels (all 37 layers), parameters = all 14 variables, dataset = interim_mnth, step = 0, version = 1, time = 00:00:00, 06:00:00, 12:00:00, 18:00:00, date = 20080901 to 20081201, grid = $1.5° \times 1.5°$, type = analysis, class = ERA Interim.

Author contributions. J. D. Silver wrote the layer-packing python software, performed the compression experiments and wrote most of the manuscript. C. S. Zender contributed to the design of the study, provided some of the test datasets used in the experiments and contributed to the text.

7 Competing interests

The authors declare that they have no conflict of interest.

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

Acknowledgements. The work of J. D. Silver was funded by the University of Melbourne's McKenzie Postdoctoral Fellowship programme. The work of C. S. Zender was funded by NASA ACCESS NNX12AF48A and NNX14AH55A and by DOE ACME DE-SC0012998. We thank Peter J. Rayner (University of Melbourne) for useful discussions.

Geosci. Model Dev. Discuss., doi:10.5194/gmd-2016-177, 2016
Manuscript under review for journal Geosci. Model Dev.
Published: 29 July 2016

References

Berger, T. (2003), Rate-distortion theory, in *Wiley Encyclopedia of Telecommunications*, John Wiley & Sons, Inc., doi:10.1002/0471219282.eot142.

Brasseur, G., D. Hauglustaine, S. Walters, P. Rasch, J.-F. Müller, C. Granier, and X. Tie (1998), MOZART, a global chemical transport model for ozone and related chemical tracers: 1. Model description, *Journal of Geophysical Research: Atmospheres*, *103*(D21), 28,265–28,289.

Caron, J. (2014), Converting GRIB to netCDF-4: Compression studies, www.ecmwf.int/sites/default/files/elibrary/2014/13711-converting-grib-netcdf-4.pdf, Last accessed: 2016-06-17, presentation to the workshop "Closing the GRIB/NetCDF gap", held at European Centre for Medium Range Weather Forecasts (ECMWF) at Reading, UK, 24-25 September 2014.

Dee, D., S. Uppala, A. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. Balmaseda, G. Balsamo, P. Bauer, and P. Bechtold (2011), The ERA-Interim reanalysis: Configuration and performance of the data assimilation system, *Quarterly Journal of the Royal Meteorological Society*, *137*(656), 553–597.

Dennis, J. M., J. Edwards, K. J. Evans, O. Guba, P. H. Lauritzen, A. A. Mirin, A. St-Cyr, M. A. Taylor, and P. H. Worley (2012), CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model, *Int. J. High Perform. Comput. Appl.*, *26*, 74–89, doi:10.1177/1094342011428142.

Deutsch, L. P. (2008), DEFLATE compressed data format specification version 1.3, *Tech. Rep. Tech. Rep. IETF RFC1951*, Internet Engineering Task Force.

Group, H. (2016), Filters, www.hdfgroup.org/services/filters.html, Last accessed: 2016-06-17.

Hübbe, N., and J. Kunkel (2013), Reducing the HPC-datastorage footprint with MAFISC – Multidimensional Adaptive Filtering Improved Scientific data Compression, *Computer Science-Research and Development*, *28*, 231–239.

Rienecker, M. M., M. J. Suarez, R. Gelaro, R. Todling, J. Bacmeister, E. Liu, M. G. Bosilovich, S. D. Schubert, L. Takacs, G.-K. Kim, et al. (2011), MERRA: NASA's modern-era retrospective analysis for research and applications, *Journal of Climate*, *24*(14), 3624–3648.

Skamarock, W. C., J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang, and J. G. Powers (2005), A description of the advanced research WRF version 2, *Tech. Rep. NCAR/TN-468 STR*, National Center For Atmospheric Research, Boulder, Colarado, USA.

Unidata (2016), The NetCDF User's Guide, *Tech. rep.*, Unidata, UCAR, Boulder, CO, USA, www.unidata.ucar.edu/software/netcdf/docs/user_guide.html, Last accessed: 2016-06-17.

Zender, C. S. (2008), Analysis of self-describing gridded geoscience data with netCDF Operators (NCO), *Environmental Modelling & Software*, *23*(10), 1338–1342.

Zender, C. S. (2016), Bit Grooming: Statistically accurate precision-preserving quantization with compression, evaluated in the netCDF Operators (NCO, v4.4.8+), *Geoscientific Model Development Discussions*, *2016*, 1–18, doi:10.5194/gmd-2016-63.

Zender, C. S., H. Bian, and D. Newman (2003), Mineral Dust Entrainment And Deposition (DEAD) model: Description and 1990s dust climatology, *J. Geophys. Res.*, *108*(D14), 4416, doi:10.1029/2002JD002775.

Ziv, J., and A. Lempel (1977), A universal algorithm for sequential data compression, *IEEE Trans. on Information Theory*, *23*(3), 337–343.

Ziv, J., and A. Lempel (1978), Compression of individual sequences via variable-rate coding, *IEEE Trans. on Information Theory*, *24*(5), 530–536, doi:doi:10.1109/TIT.1978.1055934.