TESTING INTERNALLY COMPRESSED NETCDF-4 FILE FORMAT WITHIN SMOKE-IOAPI FRAMEWORK

Roger Kwok* and Sarika Kulkarni
Air Quality Planning & Science Division, California Air Resources Board, Sacramento, CA, USA

[1]1. INTRODUCTION

Continuous advances in disk storage and CPU speeds have allowed for ever increasing file sizes in numerical modeling of geoscientific fields. However, the growth in both number and size of input files outpaces the computer power, often putting stress on the underlying files sytems. For example, CMAQ's DESID (Murphy et al 2021) allows CMAQ to use many individual emissions sectors as input, which may quickly fill up the available disk space.

To manage the storage and computational demands, files are often compressed to smaller sizes when they are not used for modeling. File compression and decompression is typically accomplished offline with common standalone tools such as GZIP (Gaille & Adler 2000) or parallel BZIP (Gilchrist & Nicholov 2021). Offline compression and decompression processes are not conducted simultaneously with modeling/simulations, such as emissions ingestion into an air quality model. As a result, the processes before and after modeling not only take up additional times, but also occupy much of the RAM and/or cache memories that often slow down computational and access speeds of other users' .

Climate studies conducted within California Air Resources Board (Zhao et al 2020) revealed that the size of netcdf-3 classic WRF output files could be reduced by ~50 % via the offline "nccopy" command (Unidata[b] 2021). Further experiment with this offline command on SMOKE emissions in classic format showed drastic size reduction by up to 99%. But in terms of disk space management and processing time, this offline approach is less efficient. To let Models-3 programs, such as SMOKE and CMAQ, access to netCDF files in a compressed format, it is desirable to enable the common program interface IOAPI (Community 2021) to read, write and compress the netCDF files on the fly.

In Section 2, we illustrate steps to compile netCDF with compression. We also indicate the key Fortran subroutines that need to be edited to accommodate this novel compression approach within IOAPI framework

File size and runtime statistics resulting from SMOKE and CMAQ runs are reported in Section 3. Insights gained from the observation are discussed in Section 4. Additional output options and environment settings required for end-users are described in the same section.

2. LIBRARY IMPLEMENTATION

Conventional IOAPI is compiled with netcdf classic without any other additional libraries. To enable IOAPI's compression capability, zlib and hdf5 need to be incorporated in the netCDF. Here are the steps needed to achieve our library implementation:

Step 1 Install zlib;
Step 2 Install hdf5;
Step 3 Install netcdf with zlib and hdf5 libraries from Steps 1 and 2;
Step 4 Install netcdf fortran based on the netcdf in Step 3;
Step 5 Compile ***edited*** IOAPI with all the libraries from Steps 1 through 4;
Step 6 Compile m3tools based on the IOAPI in Step 5.

In IOAPI, the key change lies in the subroutine "crtfil3.F90" that is called mostly by "open3.F90" subroutine. The original version only has "NF_DEF_VAR" statements each of which creates a netCDF variable. Compression was enabled by adding a "NF_DEF_VAR_DEFLATE" statement after each "NF_DEF_VAR" call. Also used by Zender (2016), the DEFLATE call is based on Deutsch's algorithm (1996), which chops the whole dataset into blocks that are individually compressed.

The new IOAPI (NC7 IOAPI hereafter) reads any netcdf format, but writes NC7 by default. Alternative output format is allowed with user specification as described in Section 4.

3. TEST CASE AND VISUALIZATION

Two cases with identical area FF10 inventory and supporting files were processed by compiling SMOKEwith different compression options ...The first case (NC3) uses SMOKE with netcdf3-classic/IOAPI libraries in which no compression capability was installed The second case (NC7) uses SMOKE with netcdf4-classic/IOAPI libraries with compression. To take storage management into account, run times to zip up NC3 files were also calculated. Two offline compression tools, GZIP and parallel BZIP (PBZIP2), were used. The resulting run times and output file sizes during SMOKE processing for an annual simulation (I.e. 365 days) of gridded hourly emissions for the two cases is shown in Table 1. First, we show that the per-day file size of NC7 was reduced to 8.7% of its NC3 counterpart, producing a 91% file size reduction. Second, we found that with respect to SMOKE run time alone, the NC7 version takes 36% longer to run than its NC3 counterpart.

SMOKE area run for 365 days					
		NC3	NC7	NC7-to-NC3 ratio	Remarks
SMOKE	Per day file size (MB)	633.0	55.0	8.7%	NC3 uncompressed
	Run time (min)	30.6	41.6	135.9%	Time taken by SMOKE
GZIP	Per day file size (MB)	55.0	N\A	100.0%	NC3 compressed
	Runtime (min)	29.6	N\A	69.2%	Time taken by SMOKE + gzip
PBZIP2	Per day file size (MB)	52.0	N\A	105.8%	NC3 compressed
	Runtime (min)	85.6	N\A	35.8%	Time taken by SMOKE + pbzip2

Table 1. SMOKE area run for 365 days. NC3 means SMOKE compiled with netcdf3-classic libraries, while NC7 means that with netcdf4-classic compressible libraries.

However, when offline compression time on NC3 files is combined with the SMOKE run time, the NC7 SMOKE still runs faster given that it compresses output files while writing them out. Compared to GZIP, NC7 SMOKE has only 69% of its combined run time; for PBZIP2, NC7 SMOKE has 36%.

Similar statistics were also observed (not shown) when a SMOKE utility called Mrggrid was also used to merge multiple emission sectors for a month. The resulting NC7 files had much smaller file size than NC3. Additionally, the NC7 SMOKE took less time to finish its run time compared to NC3 SMOKE plus offline compression combined.

Lastly, VERDI was used to verify that the output files from the NC3 and NC7 run cases resulted in identical datasets. Visualization tools such as NCVIEW, MATPLOTLIB, and VERDI, work for NC7 gridded files. In particular, VERDI not only can load NC7 and NC3 files onto the same console, but also can perform arithmetic operations on the pair to derive a third tile plot, assuming the pair have common header attributes.

4. DISCUSSIONS

While NC7 SMOKE, its utilities as well as NC7 IOAPI m3tools reduce file size tremendously, CMAQ benefits much less from it. In-house simulations indicate that CMAQ runs need longer runtimes (~ 15 –20%) with NC7 IOAPI compared to its NC3 counterpart. Also, compressing CMAQ output files such as CGRID, ACONC among others, results in only few percent reduction in file size. The contrasting behaviors exhibited by SMOKE files and CMAQ output files could be attributed to to their respective data structure. In SMOKE, data are typically filled into initialized arrays which result in discrete and sparse data structure. On the other hand, CMAQ output fields are written out after undergoing numerous multistage in-model calculations that span across grid cells and layers over the entire simulation period. As a result, all elements are

filled in the arrays thus rendering the data structure as continuous. According to Deutsch's algorithm, arrays are typically chopped into smaller blocks. The latter are then compressed individually. Arrays with sparse data structure are much easier to chop up and compressed than those with continuous structures. NC7 CMAQ performance is impacted adversely when instrumented tools such as DDM are deployed, in both file size reduction and run time.

The default output format in recent CMAQ versions is 64-bit offset, which handles files whose sizes are larger than 2 GB (Unidata[a] 2021). It is therefore desirable for NC7 IOAPI to retain this output option to allow for handling larger files when compression is less efficient. As in CMAQ's regular run scripts, the environment variable IOAPI_OFFSET_64 is reinstated in NC7 IOAPI. Table 2 shows all the environment variables recognized by the current implementation.

Environment variable	Default value	Other values	Effects
COMPRESS_NC	Y	N	➢ Y to compress output ➢ N to disable compression ➢ ***Ignored if IOAPI_OFFSET_64 Y***
USR_DFLAT_LVL	2	1 thru 9 except 2	➢ 1 mildest, 9 heaviest ➢ Ignored if COMPRESS_NC N
IOAPI_OFFSET_64	N	Y	➢ ***Y to output 64-bit offset and disable compression*** ➢ N to resume compression.

Table 2. Environment settings associated with NC7 IOAPI

COMPRESS_NC is Y by default, unless users wish to disable compression where N would be specified instead.

USR_DFLAT_LVL allows users to choose how much file size reduction is needed on the expense of processing time. 1 is the mildest but quicker and 9 the heaviest but slower. The setting will be ignored if COMPRESS_NC is set to N.

Normally, IOAPI_OFFSET_64 is N by default, letting CMAQ or other Models3 programs to go ahead with compression. If large file support is desired, users can set it to N, which disables COMPRESS_NC and ignore any settings in USR_DFLAT_LVL.

5. SUMMARY

The compressible netcdf-4/IOAPI reduces file size drastically in files with sparse data structure. This results in freeing up significant disk space without the need to do the offline compression that consumes extra system and run times. While compression is less effective on files with continuous data structure, the IOAPI still offers users an option to restore the 64-bit offset output format.

6. DISCLAIMER

NC7 IOAPI was implemented based on the official IOAPI developed and maintained by Baron Advanced Meteorological Systems (BAMS). It is not related to any policy endorsed by U.S.EPA or BAMS.

7. REFERENCES

Community Modeling and Analysis System: The EDSS/Models-3 I/O API documentation, available at https://cmascenter.org/ioapi/documentation/all_versions/html/index.html (last access November 1, 2021), -2021.

Deutsch, L. P.: DEFLATE compressed data format specification version 1.3, Tech. Rep. IETF RFC1951, Internet Engineering Task Force, Menlo Park, CA, USA, doi:10.17487/RFC1951, 1996.

Gailly, J.-L. and Adler, M.: zlib documentation, available at: http://zlib.net (last access: November 1, 2021), 2000.

Gilchrist, J. and Nikolov, Y.: Parallel BZIP2 documentation, available at http://compression.ca/pbzip2 (last access: November 1, 2021), 2003-2021.

HDF Group: HDF5: API Specification Reference Manual, The HDF Group, Champaign-Urbana,

IL, USA, 2015. Available at https://docs.hdfgroup.org/hdf5/develop/_r_m.html (last access October 5, 2021).

Murphy, B.N., Nolte, C.G., Sidi, F., Bash, J.O., Appel, K.W., Jang, C., Kang, D., Kelly, J., Mathur, R., Napelenok, S., Pouliot, G., and Pye, H.O.T.: The Detailed Emissions Scaling, Isolation, and Diagnostic (DESID) module in the Community Multiscale Air Quality (CMAQ) modeling system version 5.3.2. Geosci. Model Dev., 14, 3407–3420, 2021. https://doi.org/10.5194/gmd-14-3407-2021

Rew, R., Hartnett, E., and Caron, J.: NetCDF-4: Software implementing an enhanced data model for the geosciences, in: Proceedings of the 22nd AMS Conference on Interactive Information and Processing Systems for Meteorology, 24–28 January 2006, p. 6.6, American Meteorological Society, AMS Press, Boston, MA, USA, 2006.

Unidata Program Center[a]: Questions about netCDF large file support FAQ, available at https://www.unidata.ucar.edu/software/netcdf/faq-lfs.html (last access: November 2021)

Unidata Program Center[b]: NetCDF 4.8.0, FAQ available at: https://www.unidata.ucar.edu/software/netcdf/docs/faq.html (last access: November 1, 2021), 2021.

Zhao, Z., Di, P., Chen, S., Avise, J., Kaduwela, A., and DaMassa, J.: Assessment of climate change impact over California using dynamical downscaling with a bias correction technique: method validation and analyses of summertime results. Climate Dynamics, 54, 3705-3728, 2020. Assessment of climate change impact over California using dynamical downscaling with a bias correction technique: method validation and analyses of summertime results | SpringerLink

Zender, C.S.: Bit Grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netCDF Operators (NCO, v4.4.8+). Geosci. Model Dev., 9, 3199-3211, 2016. www.geosci-model-dev.net/9/3199/2016/doi:10.5194/gmd-9-3199-2016