# Performance Study of HDF5-WRF IO modules

MuQun Yang
Robert E. McGrath
Mike Folk
National Center for Supercomputing Applications
University of Illinois, Urbana-Champaign

## 1. Introduction to HDF5

HDF5 was developed to support contemporary high performance computing environments, including files larger than 2 Gigabytes, very large numbers of objects in a file, and alternative storage layouts [1]. HDF5 provides a number of important features, including:

- chunked storage
- external compression packages
- designed to use Message Passing Interface Input and Output (MPI-I/O)

HDF5 is available on most important high performance computing platforms.

HDF5 is used in many scientific and technical projects and communities, including important Earth Science data. The NASA EOS system provided data in HDF5 beginning with the Aura satellite [3,4]. The NPOESS system [5] uses HDF5 as one of its principle format.

Currently, Unidata and NCSA are collaborating to design netCDF4, a new netCDF built on top of HDF5 [2].

## 2. Why a WRF-HDF5 module?

WRF is designed in such a way that it has standard configuration and IO common APIs to enable external IO packages to be easily added to. An application can select which IO module to use.

The current WRF software package supports a netCDF-WRF IO module. The current netCDF IO module uses one process to collect/distribute data from other processes; then uses the same process to read/write netCDF files. This process will become IO bottleneck and the IO module may exceed the memory capacity in some applications.

To illustrate the effect of this bottleneck, the performance was measured for an example application using sequential IO. The example was run on the NCSA Teragrid linux cluster using data provided by John Michalakes of NCAR MMM Division.

In Figure 1, the wall clock time for a model run using the sequential WRF HDF5 IO module is compared for 128 and 256 processors. Experiments show that the performance of the WRF-sequential HDF5 IO module is roughly equivalent to WRF-netCDF IO module.

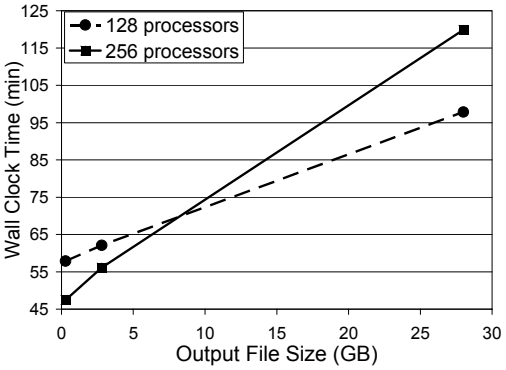


**Figure 1: Performance of Sequential HDF5 WRF Module With Different Output File Size at NCSA Teragrid Linux Cluster**

Six model runs were made; each with 900 timesteps. Three model runs were assigned 128 processors; the other 3 runs were assigned 256 processors. In the individual run, the model wrote data at every ninth timestep, every ninetieth timestep and every nine hundredth timestep. Each write is the same amount of data, so the three runs wrote a total of about 28 GB, 2.8 GB and 280 MB,respectively. In Figure 1, as the output file size (number of writes) increases, the wall clock time of the model run with 256 processors increases significantly and eventually exceeds the wall clock time of the model with 128 processors.

The reason is that it will have more communication overhead with 256 processors than with 128 processors when only one IO node can be assigned to generate output. As more output is generated, the overhead eventually exceeds the gain of the computing performance with 256 processors. The solution for this problem is to implement a parallel IO module in WRF model.

Since this example is a real WRF application; it shows that the parallel IO module is necessary to be incorporated into the WRF model.

For some WRF applications, the output is huge. In-memory data compression can save storage, network bandwidth, and often improves IO performance as well. TheWRF-sequential netCDF module does not provide in-

memory compression support. Currently HDF5 supports various compression packages; therefore it is possible for application users to adopt compression features with WRF-sequential HDF5 module.

### 3. Characteristics of Sequential and Parallel HDF5-WRF Design and File Structure

There are two HDF5 WRF IO modules, sequential and parallel. The former uses the serial HDF5 library, and is very similar to the standard netCDF WRF IO module. The sequential HDF5 module supports data compression.

The Parallel HDF5-WRF uses the parallel HDF5 library, which is implemented with MPI IO. In contrast to the sequential module, every computing node will also be used for IO through Parallel HDF5. This provides high performance, especially for large data writes and reads.

In both cases, the file structure of the HDF5 output is similar to that of NetCDF to help NetCDF users easily understand the WRF-HDF5 IO modules.

For more information about the design and prototype implementation, see [7].

### 4. Performance report

The performance of the WRF-parallel HDF5 IO module was tested on several platforms. The performance study of WRF-Parallel HDF5 module provides a guideline for users to adopt parallel options with NetCDF4 in the future.

Three real WRF cases are used for performance study.

- Case 1: A real weather forecasting case. The maximum hyperslab size is about 17 MB per time step.
- Case 2: A real short-range climate forecasting case. The maximum hyperslab size is 3 MB per time step.
- Case 3: A numeric simulation case of a squall line. The maximum hyperslab size is 1.9 MB per time step.

The platforms included the NCSA and SDSC Teragrid Linux clusters; NCAR IBM power3, and the NCSA IBM power 4. The wall-clock time to complete a model run is compared for parallel HDF5-WRF and sequential netCDF modules. All model settings are exactly the same except for the output data.

#### 4.1. Parallel HDF5 WRF IO Module

Figure 2 shows a comparison of the wall clock time between WRF-parallel HDF5 IO module and WRF-sequential netCDF IO module for Case 1 at NCAR IBM with 256 processors. The maximum hyperslab size per timestep is 17 MB. The WRF-parallel HDF5 IO module outperforms WRF-netCDF IO module as the output file size increases. When the output file size reached around 20GB, the model run took about 80 minutes wall clock time when using WRF-sequential netCDF IO module; compared to 20 minutes when using parallel WRF-Parallel HDF5 IO module.

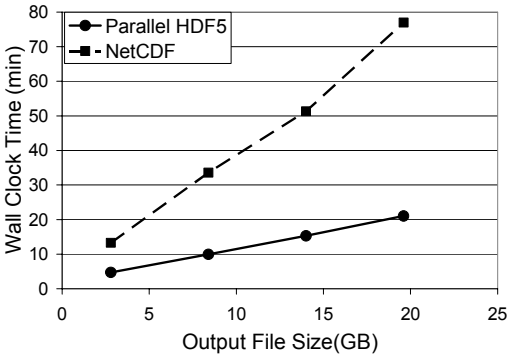


**Figure 2: Performance of Parallel HDF5 to sequential NetCDF for Different Output File Size IBM Power 3 (256 Processors) (Case 1)**

Similar experiments with Case 2 (a real short-range climate forecasting case) gave similar results.

However, in some cases, WRF-sequential netCDF IO outperforms the WRF-parallel HDF5 IO module. Figure 3 shows results for case 3 at NCSA IBM Power 4 with 16 processors. The maximum hyperslab size per timestep in this case is 1.9 MB, only one-eighth the size of that in case 1. The performance of the WRF-parallel HDF5 is consistently worse than that of WRF-sequential netCDF. In this case, the overhead of MPI-IO outweighs the gain from multiple processors.

Other analysis shows that when case 3 is run with 64 processors at NCAR IBM, WRF-parallel HDF5 is slightly better than WRF-sequential netCDF. So the performance of WRF-parallel HDF5 appears to improve with the increasing of number of processors assigned to it.

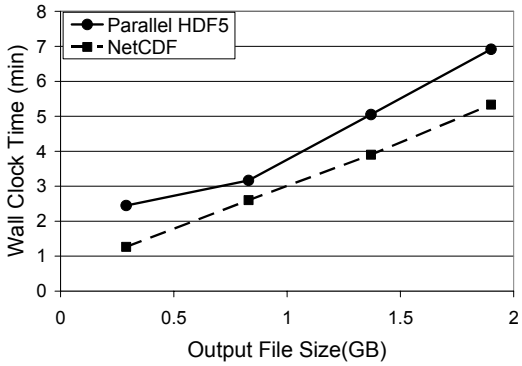


**Figure 3: Performance of Parallel HDF5 and sequential netCDF with Different Output File Size IBM Power 4 (16 processors) (Case 3)**

However, increasing the number of processors does not always improve the performance of parallel IO. In another experiment, the baroclinc wave case provided by WRF was run with different numbers of processors at San Diego Supercomputer Center Teragrid Linux cluster. Two processors are assigned for each node. The maximum hyperslab size is about 4.7MB, and the model was run with 4-64 processors. Figure 4 shows that with the increasing of number of processors, the performance of WRF-parallel HDF5 decreases.

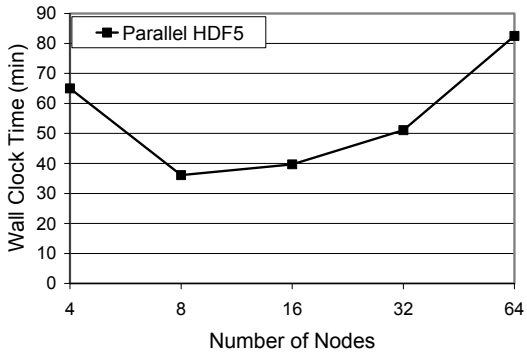


**Figure 4: Performance of parallel HDF5 with the same Output File Size but Different Number of Processors Teragrid Linux cluster (2 processors per node); baroclinic wave case**

The following summarizes our findings related to this issue and we hope that will help readers who will use parallel IO in the future.

- Parallel HDF5-WRF IO module can greatly improve WRF IO performance for **some** WRF applications.
- The parallel IO performance will depend on the size of the HDF5 dataset, the number of processors, the platform used, the parallel file system, MPI-IO library.
- According to some experiments, it appears to be the larger the HDF5 dataset size is and the more processors the WRF can run, the better the IO performance can be possibly achieved on IBM SP GPFS.

### 4.2. Sequential IO with Data Compression

The sequential HDF5 module supports in-memory compression. The HDF5 library supports SZIP and deflate (GZIP) compression algorithms, as well as an optional byte shuffle algorithm that can improve the compression [8] [9]. This study compared three storage options:

- WRF-sequential HDF5 without compression,
- WRF-sequential HDF5 with SZIP compression, and
- WRF-sequential HDF5 combined with shuffle plus GZIP compression.

For each option, the model was run for 10, 30, 50 and 70 timesteps. For each timestep, the model wrote the HDF5 output into the same HDF5 history file.

Figure 5 shows file size of different runs for each option. As expected, the overall HDF5 file size in either SZIP compression or shuffle with GZIP compression is much less than the same data without compression. For example, for the data at timestep 70, the file size is about 2000 MB without compression, compared to less than 600MB for either SZIP or GZIP compression.

Figure 6 shows the overall wall clock time of different runs for each option. The wall clock time to write the data with SZIP compression is just a little greater than that without compression. The wall clock time to write the data with shuffle and deflate compression is significantly greater than that without compression. For example, it took about 4 more minutes to compress 2000MB data. SZIP is much faster when compressing the WRF data.

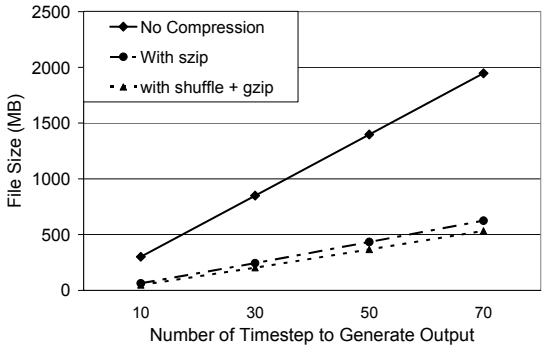


**Figure 5: Performance of Sequential HDF5 with Different Compression Methods (Case 3: Squall Line) IBM Power 4 (16 processors)**

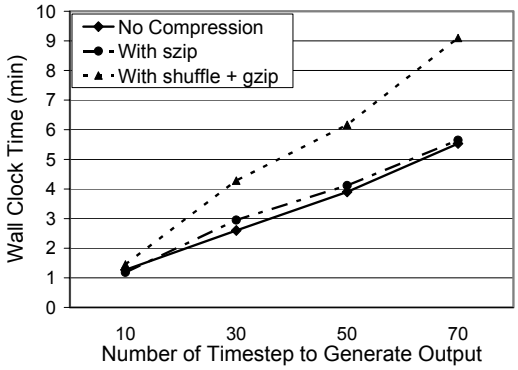


**Figure 6: Model Output File Size With Different Compression Methods (Case 3: Squall Line) IBM Power 4 (16 processors)**

The results of the compression study show that:

- SZIP compression can greatly compress the WRF data with little encoding time overhead.

- The Shuffle algorithm combined with deflate compression can greatly compress the WRF data with some encoding time overhead.

**5. Where to obtain the HDF5 WRF IO Modules**

WRF-Parallel HDF5 and Sequential HDF5 IO modules can be obtained at:
http://hdf.ncsa.uiuc.edu/apps/WRF-ROMS/

**References**


1. http://hdf.ncsa.uiuc.edu/HDF5
2. http://www.unidata.ucar.edu/proposals/NASA-AIST-2002/
3. http://www.eos.ucar.edu/hirdls/
4. http://lennier.gsfc.nasa.gov/seeds/W3pr_Craig.pdf
5. http://www.ipo.noaa.gov/
6. http://www.ncsa.uiuc.edu/expeditions/MEAD/
7. http://www.ncsa.uiuc.edu/apps/WRF-ROMS
8. http://www.ncsa.uiuc.edu/HDF5/papers/
9. http://hdf.ncsa.uiuc.edu/doc_resource/SZIP/



**Acknowledgements:**

Authors want to thank John Michalakes at National Center for Atmospheric Research for providing CONUS WRF initial data, Dr. Xinzhong Liang at Illinois State Water Survey for the Climate WRF initial data, Dr. Brian Jewett at Dept. of Atmospheric Sciences at University of Illinois at Urbana-Champaign for the squall line WRF initial data.

This work is part of NSF-funded Modeling Environment for Atmospheric Discovery Expedition and we appreciate the timely supports through NSF-sponsored TeraGrid project.