WRF-HDF5 IO performance report

MuQun Yang, Mike Folk

NCSA HDF Group
October 1, 2003

Abstract

This paper presents the WRF-HDF5 parallel IO performance results funded by NSF Alliance MEAD project. It shows that the HDF5 file structure has to be changed for parallel HDF5 IO module. Three real WRF cases are used to evaluate the WRF-HDF5 parallel IO performance. The results show that parallel WRF-HDF5 module can greatly reduce wall clock time in some WRF applications. The performance largely depends on the size of the WRF output array at each time step and the number of processors assigned to the model run.

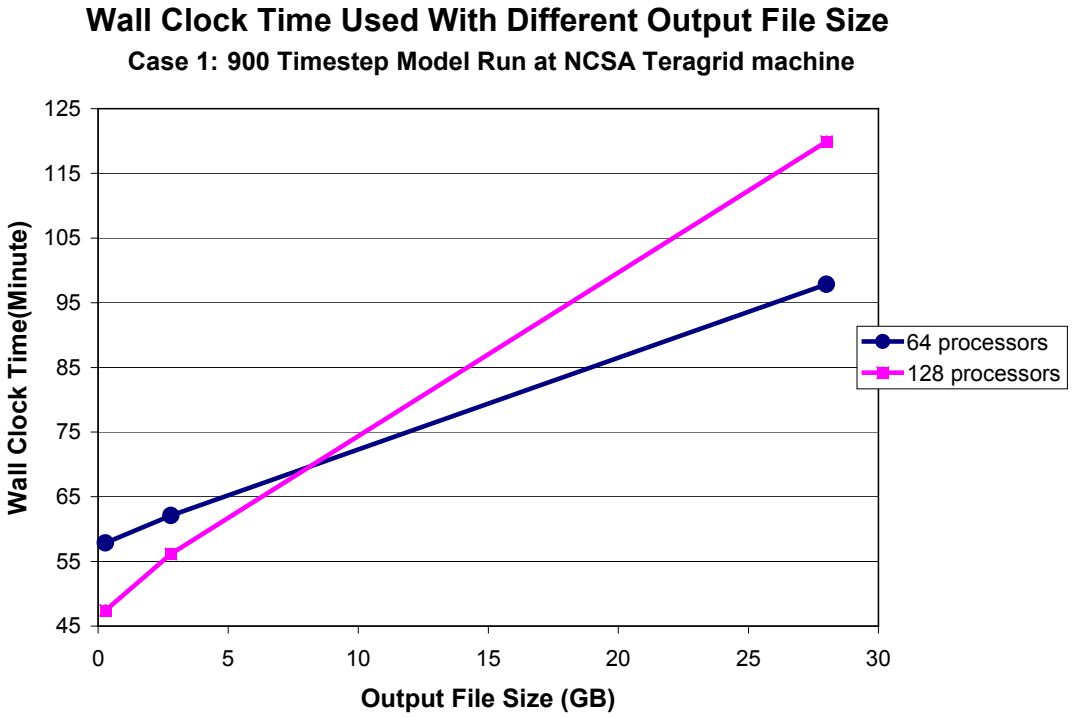
Contents

I. Why we need Parallel-IO module in WRF model

In the previous report [1], we addressed the importance of a parallel-IO module in WRF model. Here we will use a real WRF case to show the importance of a parallel-IO module inside WRF model.

The IO module we are using is the sequential WRF-HDF5 IO module. Experiments show that without using compression, the IO performance of the sequential WRF-HDF5 IO module is roughly equivalent to WRF-NetCDF IO module. The example case was run on the NCSA Teragrid machine (Mercury). The data are provided by John Michalakes of NCAR MMM Division. The detailed information of the data is described at section III. We set the total computing time step to 900 and then make 6 model runs. We assign 128 processors for 3 model runs and 256 processors for other runs. In the individual run we let the model output data at every 9 timestep, every 90 timestep and every 900 timestep. This is roughly equivalent to file size of 28 GB, 2.8 GB and 280 MB. The figure shows the change of the wall clock time the model used corresponding with different output file size.

Fig. 1



We can observe that with the increasing of the output file size, the wall clock time of the model with 256 processors catches up and eventually exceeds the wall clock time of the model with 128 processors; which is unexpected at a glance. This means: you expect to spend more CPU resources hoping to get your result back soon; but on the contrary, it takes much longer for the results to get back. The reason is simple: 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 becomes even severe and eventually it will overcome 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.

II. Introduction to parallel WRF-HDF5 IO module

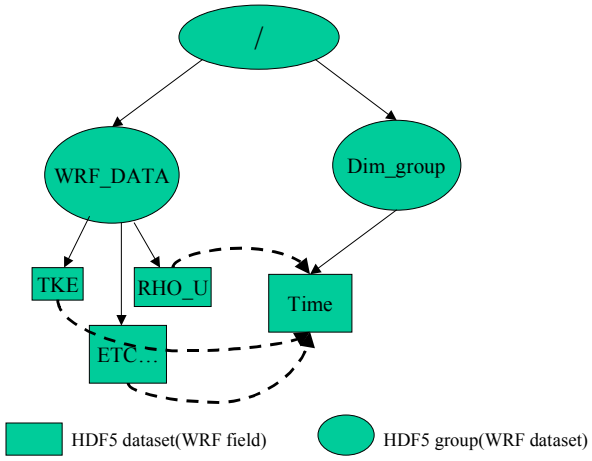
1) Design

See our previous report "Investigation of parallel IO module in WRF"[1]. For the purpose of MEAD project, we decide to use every computing node also as an IO node to generate output via parallel HDF5.

2) File structure

Originally we would like to keep the same file structure as the sequential WRF-HDF5 IO module. In order to make the WRF-HDF5 dataset extensible, chunking storage inside parallel HDF5 has to be used. However, after some experiments, we found that the current parallel HDF5 version with chunking storage cannot improve WRF IO performance. We then decide to use parallel HDF5 with the contiguous storage for WRF. To do this, we have to change the file structure of the HDF5 file. The sequential WRF-HDF5 file structure is shown at figure 2 and the parallel WRF-HDF5 file structure is shown at figure 3. The main difference is that the data output of each time step will be under one group for parallel file structure so that one can still extend the dataset by adding more groups.

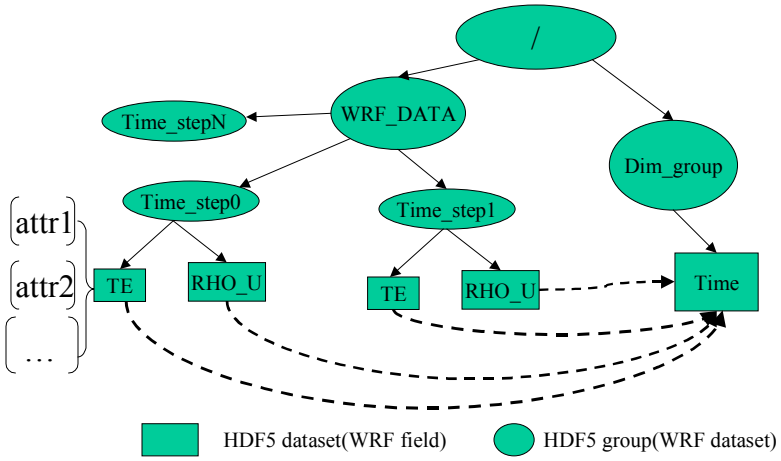
Fig. 2: Schematic HDF5 File Structure of Sequential HDF5-WRF Output



Solid line: HDF5 datasets or sub-groups (the arrow points to) that are members of the HDF5 parent group.

Dash line: The association of one HDF5 object to another HDF5 object through dimensional scale table.

Fig.3: Schematic HDF5 File Structure of Parallel HDF5-WRF Output



Solid line: HDF5 datasets or sub-groups (the arrow points to) that are members of the HDF5 parent group.

Dash line: the association of one HDF5 object to another HDF5 object through dimensional table.

3) Limitation and future work

The current HDF5 file structure of the parallel IO module is not consistent with that of the sequential IO module. This should be avoided because that will cause the post-processing of data output difficult. The HDF5 file structure of the sequential IO is more consistent with the corresponding NetCDF file structure; which is widely accepted by the WRF user community. In the future, we should change back to the HDF5 file structure in Fig.2 when HDF5 library improves the parallel IO performance with chunking storage. The parallel IO performance metrics on the following report will still be based on the contiguous storage in the current Parallel HDF5 version and the output file structure in Fig. 3. Regardless of the data storage method, the same amount of data should be written into the disk anyway; so the performance report in a sense is still a good guideline for the future parallel HDF5-IO module of WRF.

III. Data Source

1) Case 1(conus case):

This is a real weather forecasting case from Oct. 24th, 2001 to Oct. 25th, 2001. The maximum array size is about 17 MB per time step.

2) Case 2(Climate WRF case):

This is a real short-range climate forecasting case from May 2nd, 1993 to June 1st, 1993. The maximum array size is 3 MB per time step.

3) Case 3(Squall line case):

This is a numeric simulation case of a squall line. The maximum array size is 1.9 MB per time step.

IV. Performance analysis

Since we need a real parallel file system to evaluate parallel IO performance; two machines are used for the performance analysis. One is IBM p690 (Regatta) at NCSA and another is IBM power 3(WinterHawkII) at NCAR. We use 16 processors or 8 processors at Regatta and up to 256 processors at WinterHawk II to run WRF.

1) Case 1(conus case):

EM core of WRF is used to run this case. Fig. 4 and Fig.5 show the performance comparison between Parallel HDF5 and NetCDF IO module. The X-axis is the output file size and the y-axis is the wall clock time. Either in Regatta or in WinterHawk II, the parallel HDF5 always performs better than NetCDF. With the increasing of the number of processors and the increasing number of File Size (compared Fig.4 with Fig5), parallel IO module becomes even more efficient.

Fig. 4: Wall Clock Time Used at Different Output File Size
Case 1: Conus
IBM Regatta (16 processors)

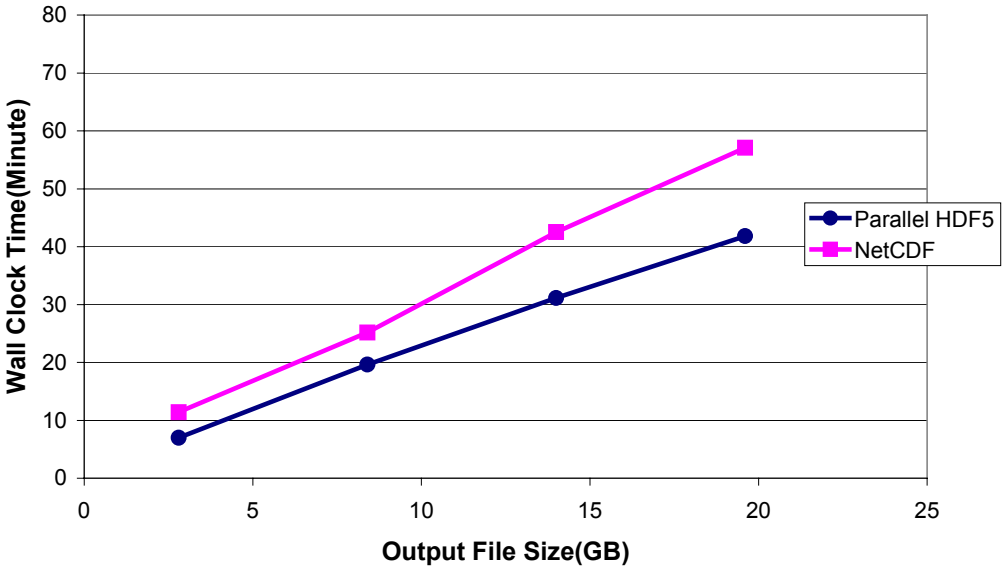
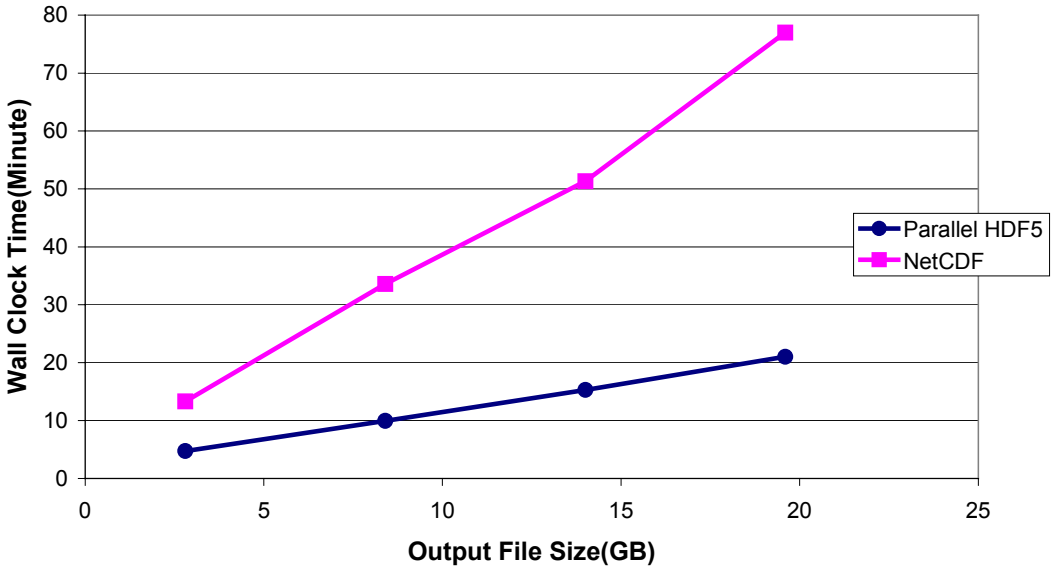


Fig.5: Wall Clock Time Used with Different Output File Size
Case 1: Conus
IBM WinterHawkII (256 Processors)



2) Case 2(CWRF case):

Fig. 6 shows the performance comparison between Parallel HDF5 and NetCDF IO module for this case with 8 processors in IBM Regatta. Fig. 7: with 32 processors in IBM WinterHawk II. EH core of WRF is used to run this case. One can find that the performance of Parallel HDF5 in Fig.6 is worse than that of NetCDF IO module! That's kind of surprising at the first glance. However, when one realizes that it needs extra overhead to set up parallel IO, the overall performance depends upon the net effect of the reduction of CPU time through the data written into disks in parallel and extra time caused to set up the parallel IO environment (extra system calls, library API calls). You may notice that the maximum array per time step in this case is almost 6 times smaller than that in Case 1 and this will cause writing the data in disk less efficient; but the overhead to set up the parallel IO environment may be the same. So the net effect is that there will be more wall clock time used for parallel IO module that sequential IO module for this case. However, one way to increase parallelism is to use more processors and that should theoretically increase the parallelism and it may turn out that the overall performance becomes better again. Fig. 7 exactly showed this case. When using 32 processors to run WRF, we find that the performance of Parallel WRF-HDF5 IO module again exceeds that of NetCDF IO; not as dramatically as case 1.

Fig.6: Wall Clock Time Used With different Output File Size
Case 2: Climate WRF
IBM Regatta (8 Processors)

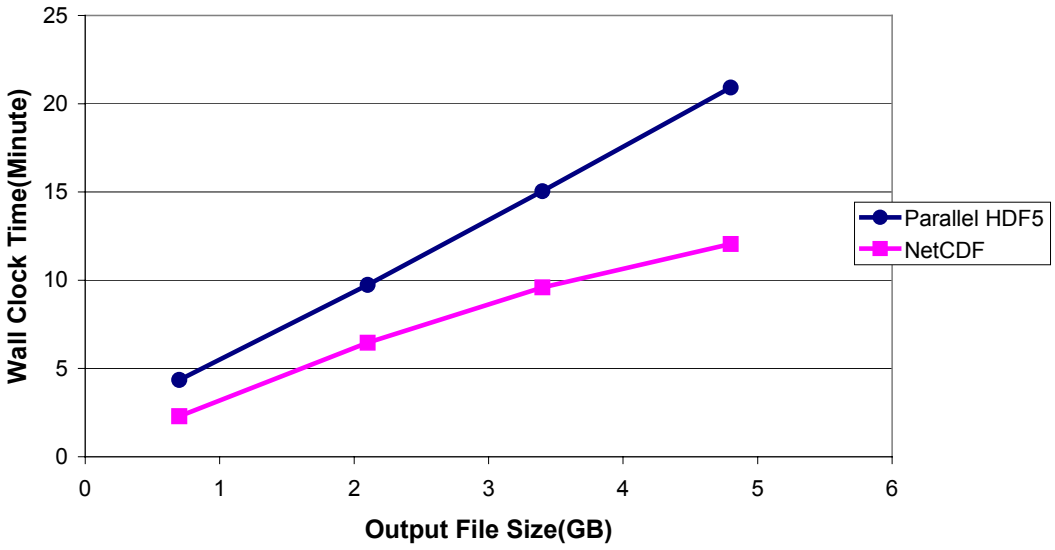
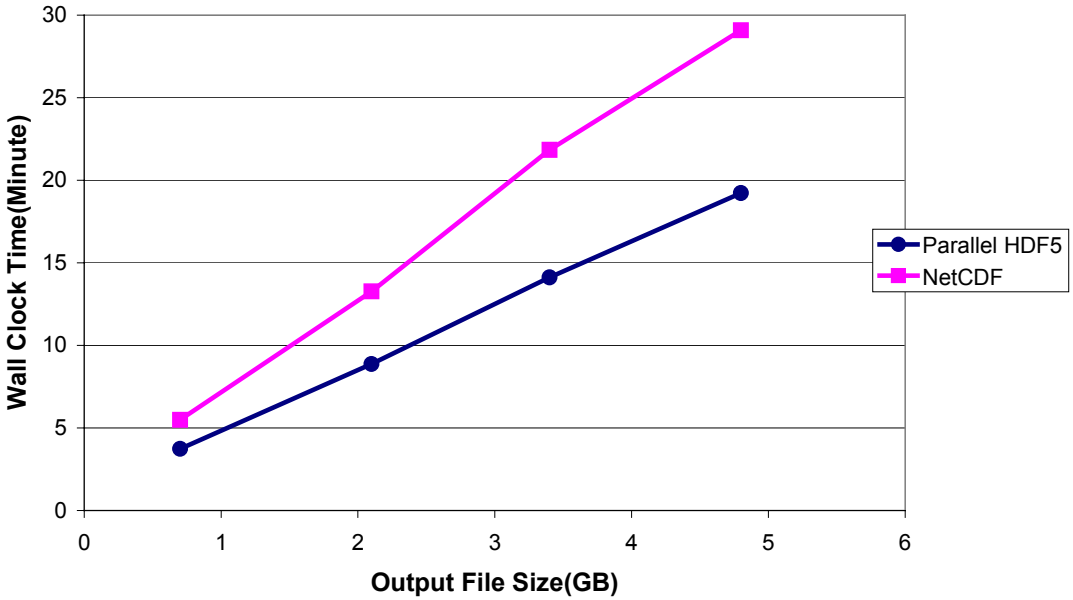


Fig.7: Wall Clock Time Used With Different Output File Size
Case 2: Climate WRF
IBM WinterHawkII (32 processors)



3) Case 3(squall-line case):

Case 3 is very similar to Case 2. The difference is that it used EM core. The maximum array size per time step is 1.7 MB, a little smaller than that in case 2. Fig. 8 and Fig. 9 shows similar results as those in Case 2. One result not shown is that parallel IO won't assure you to always gain better performance as the number of processors assigned to the model increases. In this case, when we use 256 processors to run WRF, we find the wall clock time with parallel HDF5 IO module exceeds the wall clock time with NetCDF. For the smaller dataset this is expected since the reduction of wall clock time through writing data to the disk in parallel may be overcome by the overhead to set up parallel environment as there is very small portion of data written in parallel; the parallelism is not efficient any more.

Fig.8: Wall Clock Time Used With Different Output File Size
Case 3: Squall line
IBM Regatta (16 processors)

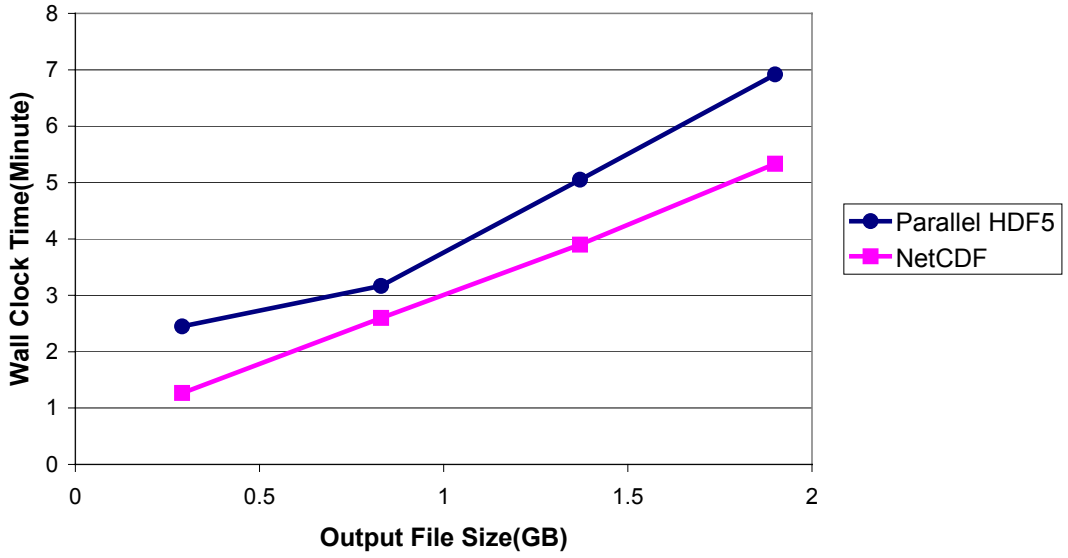
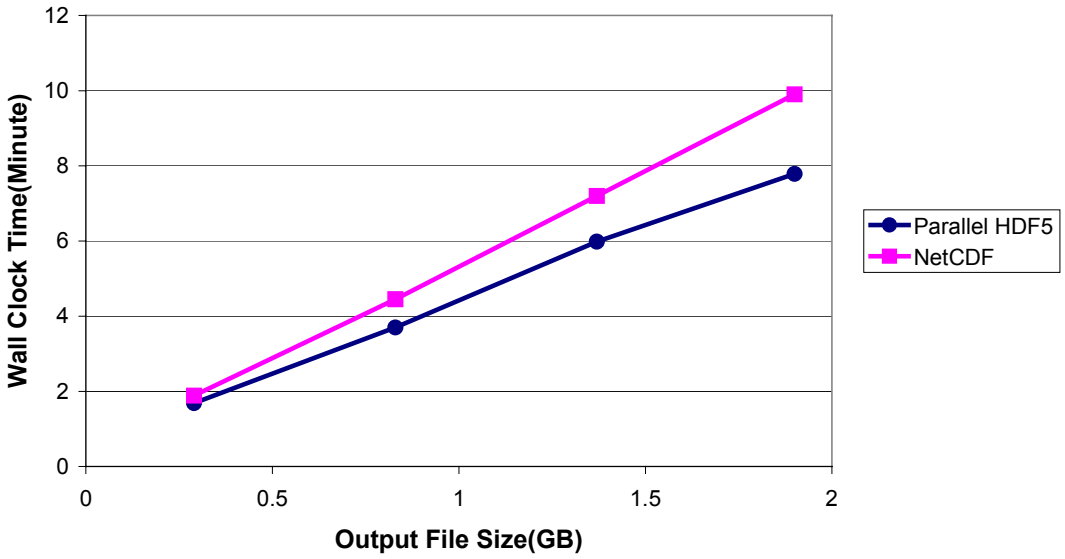


Fig.9: Wall Clock Time Used With Different Output File Size
Case 3: Squall line
IBM WinterHawkII (64 processors)



4) Summary

- Case 1 to 3 show that Parallel WRF-HDF5 module can greatly reduce wall clock time in some WRF applications.
- Since parallel WRF-HDF5 IO module handles the data output in series of hyperslab(the output data array in one time step) ; so usually the better parallel WRF-HDF5 IO performance may be achieved with the larger hyperslab size.
- The parallel WRF-HDF5 IO performance often becomes better when the number of processors assigned to the model run increases although it is not always true.
- Parallel IO module will not always improve the IO performance. Users should examine their problems carefully; especially the size of the hyperslab; the knowledge of the system and the number of processors to be used to run the job.
- To use parallel HDF5, one's machine must support parallel file system.

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.

Reference:

1. MuQun Yang, Mike Folk, *Investigation of parallel IO module in WRF*, May 8, 2003 http://hdf.ncsa.uiuc.edu/apps/WRF-ROMS/