Efficient parallel I/O with ZioLib in Community Atmosphere Model (CAM)

Yu-Heng Tseng and Chris Ding (yhtseng@lbl.gov and chqding@lbl.gov)

Computational Research Division, Lawrence Berkeley National Laboratory

Abstract

Serial I/O within a parallel environment has the potential bottleneck as the resolution increases or the output is more frequent. The maximum I/O rate on a distributed array is reached while the parallel decomposition is consistent with the last array dimension ("Z"-decomposition). Since this is not the case for arrays in CAM, remapping arrays into a Z-decomposition becomes important in order to increase I/O bandwidth. Further, history variables are stored in a disk file in a different index order than the one in CPU resident memory due to different parallel decompositions and dynamic cores. To facilitate efficient and flexible I/O in CAM, we combine the Parallel NetCDF library with ZioLib algorithm. This procedure remaps distributed arrays into a Z-decomposition on a subset of processors, and then writes to a disk file in parallel to obtain the maximum parallel performance. For a 1.1GB standard output field of CAM3.1 D-resolution run, the current procedure can speed up history I/O by a factor of over 13 on an IBM SP.

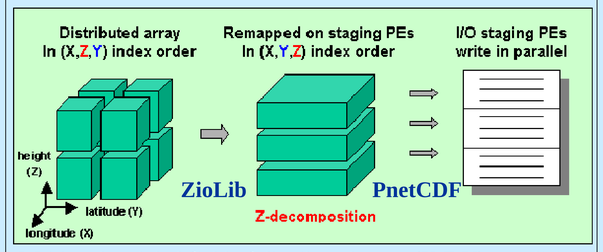
Introduction

High resolution Century- or millennium-long global climate simulations using the NCAR Community Climate System Model (CCSM) generate tremendous amount of data. Efficient I/O is a crucial factor for such large-scale simulations on massively parallel machines, but CCSM currently uses sequential I/O through a single processor. This will soon become a major bottleneck for even higher resolution simulations. Here we implement the Parallel NetCDF (PnetCDF) with ZioLib algorithm in the Community Atmosphere Model (CAM), a component of CCSM, to facilitate efficient and flexible parallel I/O.

In distributed memory parallel environment, most applications rely on a serial I/O strategy, where the global array is gathered on a single processor and then written out to a file. The I/O performance with this approach is largely limited by single PE I/O bandwidth. Even when parallel I/O is used, satisfactory parallel scaling is not always observed. Parallel I/O rates can depend sensitively on parallel decompositions. The current approach, combining the features of PnetCDF and ZioLib, ensures the flexibility and the maximum I/O rate for all physical decompositions. Our tests show that this approach greatly Improves the CAM I/O performance and removes the I/O bottleneck. The maximum speed-up is roughly scaled with the increasing domain size.

Parallel NetCDF library with ZioLib algorithm

NetCDF is a simple and widely used file format. The PnetCDF interface facilitates an efficient parallel I/O to access a single netCDF file [1]. However, the I/O rate may not be optimal depending on the array's index order and the results may not be consistent with the serial I/O. On the other hand, ZioLib [2] can flexibly remap the distributed array and output data in parallel. Thus, our strategy is to remap a distributed field into a Z-decomposition on a subset of processors ("I/O staging processors") using the ZioLib algorithm, and then write to a disk file using PnetCDF (see below figure). In this Z-decomposition, the data layout of the remapped array on the staging processors' memory is the same as on disk, thus only block data transfer occurs during parallel I/O, achieving maximum efficiency.



Writing the global field of distributed array (X,Z,Y) to a disk file in (X,Y,Z) order using three I/O staging processes.

Advantages of PNetCDF with ZioLib algorithm

- **Relieve memory limitations on a processor**
- **Relieve congestion on I/O server nodes**
- **Write/read in large blocks (no seeks) in parallel with maximum flexibility**
- **Achieve robust and maximum I/O performance regardless of parallel decomposition**
- **Eliminate a temporary global field from user codes (gather/scatter/transpose)**

PNetCDF and ZioLib Source Codes

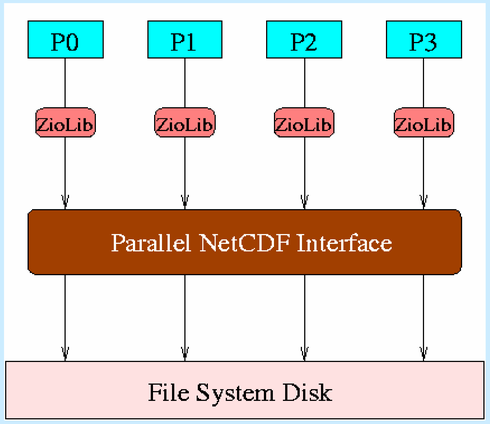
The source codes and information about the Parallel NetCDF (PnetCDF) can be downloaded from http://www.unix.mcs.nanl.gov/parallel-netcdf/.

The source codes, test examples, and user's manual of ZioLib are all available from http://crd.lbl.gov/~cding/acpi/ZioLib/.

Performance on CAM3.0/3.1 History I/O

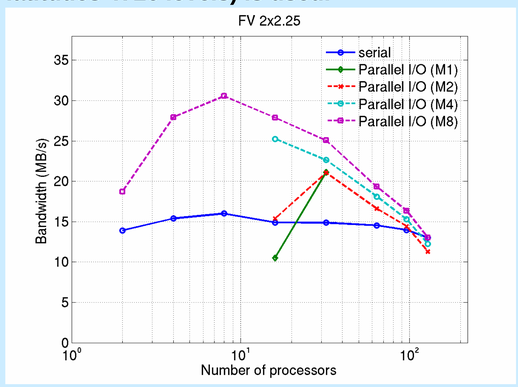
In the current CAM, a field in CPU resident memory is in one index order but is stored in a disk file in another order. For example, history data for CAM's dynamic variables are in the (longitude, height, latitude) order but must be stored in a file in the (longitude, latitude, height) order. Changing index orders complicates a parallel I/O implementation and slows down I/O.

To optimize the I/O performance, we have implemented the PnetCDF with ZioLib algorithm on CAM3.0 and CAM3.1 history I/O and compared with the serial netCDF I/O (i.e., one staging processor) using 2 to 512 MPI tasks on the IBM SP and the Linux cluster at LBNL/NERSC. All dynamic cores (Eulerian, Semi-Lagrangian and Finite Volume) are tested. The parallel implementation in CAM is illustrated below.



- **Finite Volume dynamic core**

Standard low resolution (144 longitudes x 91 latitudes x 26 levels) is used.

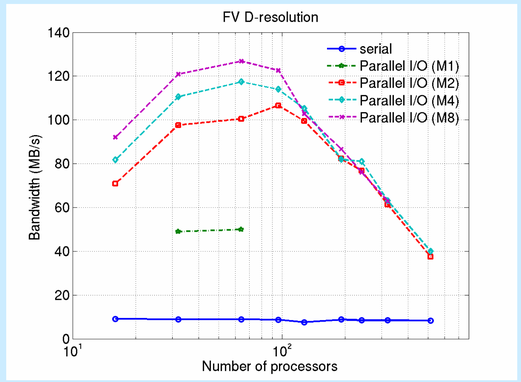


In CAM, processor 0 gathers distributed data, transposes the global array, and writes to a file. Compared to this method, the current approach speeds up by a factor of 2.

Performance of parallel I/O (continued)

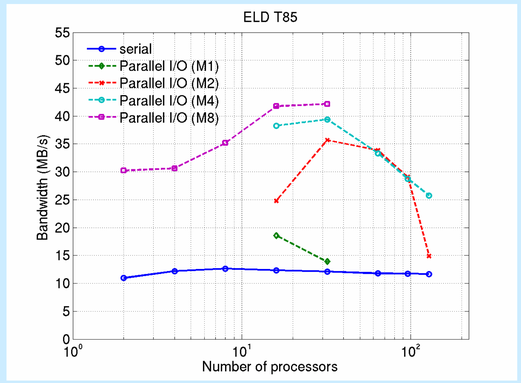
- **Same experiments for 576x361x26 D-resolution.**

Speeds up writes by a factor of over 13 with respect to the single-PE I/O. The speed-up is roughly scaled with the global domain size.



- **Eulerian dynamic core**

T85L26 resolution (256 longitudes x 128 latitudes x 26 levels) is used. The current approach speeds up by a factor of 3.



Estimated time for 20 years simulation of D- resolution (FV) on SEABORG (assuming weekly standard output)				
Procs	CAM 3.1 (Serial I/O)		CAM 3.1 (Parallel I/O)	
	Total	Serial I/O	Total	Parallel I/O
32	593 days	53 days	544 days	4 days
64	480 days	53 days	431 days	4 days

The table shows that, the speed-up of our approach is significant with respect to the existing method in larger domain size. The potential I/O bottleneck problem is greatly reduced.

Acknowledgements

This work is supported by a DOE SciDAC climate project and a NERSC Program.

Reference

[1] J. Li, W. K. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel netCDF: a high-performance scientific I/O interface", SC'2003 Nov. 15-21, 2003, Phoenix, Arizona, USA.

[2] W. S. Yang and C. Ding, "ZioLib: A parallel I/O library", Lawrence Berkeley National Laboratory Report 53521.