Implementation of parallel NetCDF in the ParFlow hydrological model: A code modernisation effort as part of a big data strategy

Lukas Poorthuis1,2, Klaus Goergen1,2,3, Wendy Sharples1, Stefan Kollet4

1. Background and motivation

State-of-the-art geoscience simulations are tending towards ever increasing model-complexity. Consequently, theUnable to load images. Please load the image again or provide an alternative.

2. ParFlow model system

(integrated parallel watershed model, fully coupled dynamic 2D/3D hydrothermal, biogeochemical, and land-surface processes. It is written in C and FORTRAN and parallelised with MPI.

It supports:
• 2D/3D saturated subsurface flow and energy transport
• Integrated overland flow and saturated subsurface flow
• Part of fully coupled multiphysics TerraNPP via external OASIS-3 MCT interface
• Usage across spatial scale from catchment to continent

3. HPC system: JSC/JURECA

JURECA configuration (2015):
• 1,872 nodes, 45,236 cores
• Node: 24 cores with up to 48 threads
• Main memory: 212 TB
• Peak performance: 1.8 TB (CPU) + 0.44 (GPU) PFLOPS
• Interconnect: InfiniBand non-blocking fat tree (CPU) + InfiniBand + 6x2690 v3.2 SGII
• RAM: 128 GB DDR4 RAM per node

4. Parallel I/O libraries, implementation and usage

Both plParCF and NetCDF provide high-performance parallel I/O.

The proper and efficient usage of the performance tuning features constitutes the challenge of these libraries. Especially the chunking feature, a process of storing multidimensional data in rectangular chunks to speed up I/O access, can offer substantial performance improvements for multi-dimensional variables.

5. Chunking with NetCDF4

The advantages of chunking with NetCDF4 (i.e., data structure of the NC-file during write-access) are a higher I/O performance when specific spatial subsets or time ranges are read from a multidimensional dataset.

Choosing the correct chunk size and respecting the access patterns of 3D and 2D data is most important.

Tab. 2: Chunk size results, i.e., input to the NetCDF-API for an arbitrary 2D variable with the dimensions 80*1000 (x axis) * 200 (y axis). The size of the data elements is assumed to be 4 B. The table lists the chunk sizes that were found optimal for the used file size. The last column lists the chunk size that was found optimal for the used file size.

6. Parallel I/O tests using the JUBE2 environment

JUBE is an essential tool that allows one to encode and document a variety of tests quickly and easily, in order to benchmark code changes and assist with the ongoing code development... JUBE is a code benchmarking tool. Originally the JUBE framework was just used to conduct the IO scaling tests and was then further extended to also support the ParFlow code base.

The currently implemented JUBE framework for ParFlow supports:

• Real data/idealised tests cases
• Healing scaling experiments
• Customisable compiler options
• Common HPC profiling tools (e.g., Score-P/Saleca, Paraver, and Darden I/O profiling)

7. I/O scaling test

This scaling study was conducted with ROMIO NetCDF to enable collective buffering for the MPI I/O library, disable the default round trip and set the collective buffering cache appropriate for the COSMO file system.

The results from the study shows that both libraries demonstrate good scaling behavior. However, NetCDF shows more consistent read performance and has a more linear scaling behavior than plParCF.

8. NetCDF4 pI/O implementation into ParFlow

To enable I/O handling with one I/O stream per node the data management of ParFlow has to be extended. To gather the data, an additional MPI communicator has to be introduced, which only communicates on the specific hardware node.

3. Overview

• Implementation of a MPI-parallel data management to only have one data stream per node (with several MPI tasks per node)
• Replacement of the ParFlow binary output module with a NetCDF I/O module
• ParFlow I/O optimisation with the profiling tools
• Compression of NetCDF output

Implementation of in situ processing in ParFlow using VisIt on JURECA to reduce total processing time and model output data volumes

Acknowledgements

References


