Geosci. Model Dev. Discuss., 7, 7427–7459, 2014
www.geosci-model-dev-discuss.net/7/7427/2014/
doi:10.5194/gmdd-7-7427-2014

Geoscientific Model Development Discussions
Open Access

An approach to enhance pnetCDF performance in environmental modeling applications

D. C. Wong[1], C. E. Yang[2], J. S. Fu[2], K. Wong[2], and Y. Gao[2,*]

[1]U.S. Environmental Protection Agency, Research Triangle Park, NC, USA
[2]University of Tennessee, Knoxville, TN, USA
[*]now at: Pacific Northwest National Laboratory, Richland, WA, USA

Received: 2 September 2014 – Accepted: 6 October 2014 – Published: 6 November 2014

Correspondence to: D. C. Wong (wong.david-c@epa.gov)

Published by Copernicus Publications on behalf of the European Geosciences Union.

Abstract

Data intensive simulations are often limited by their I/O performance, and novel techniques need to be developed in order to overcome this limitation. The software package, pnetCDF which works with parallel file systems, was developed to address this issue by providing parallel I/O capability. This study examines the performance of a novel approach which performs data aggregation along either row or column dimension of MPI processes on a spatially decomposed domain, and then applies the pnetCDF parallel I/O paradigm. The test was done with three different domain sizes which represent small, moderately large, and large data domains, using a small-scale Community Multiscale Air Quality model (CMAQ) mock-up code. The examination includes comparing I/O performance with traditional serial I/O technique, straight application of pnetCDF, and the data aggregation along row and column dimension before applying pnetCDF. After the comparison, "optimal" I/O configurations of this novel approach were quantified. Data aggregation along the row dimension (pnetCDFcr) works better than along the column dimension (pnetCDFcc) although it may perform slightly worse than the straight pnetCDF method with a small number of processors. When the number of processors becomes larger, pnetCDFcr out performs pnetCDF significantly. If the number of processors keeps increasing, pnetCDF reaches a point where the performance is even worse than the serial I/O technique. This new technique has also been tested for a real application where it performs two times better than the straight pnetCDF paradigm.

1 Introduction

The Community Multiscale Air Quality (CMAQ) model (Byun and Schere, 2006) is a regional air quality model which is widely used in air quality research and regulatory applications (e.g. Fu et al., 2012). This model was developed in the 1990s by the U.S. Environmental Protection Agency (US EPA) and it has continued to evolve. Recently, CMAQ was combined with WRF to form a WRF-CMAQ two-way coupled model (Wong et al., 2012) with direct aerosol effects on radiation. CMAQ has been and continues to be extensively used to provide guidance in rulemaking such as CSAPR (Cross-State Air Pollution Rule, http://www.epa.gov/airtransport/CSAPR/), used by state and local agencies for air quality management analyses such as SIP (State Implementation Plan), and also used by academia and industry for studying relevant atmospheric processes and model applications. CMAQ has also been adapted into the real-time US National Air Quality Forecasting system (AQF) (Otte et al., 2005) operationally at National Weather Service since 2003 and was recently deployed for forecasting air quality for the 2010 Shanghai World Expo.

CMAQ uses IOAPI (Input/Output Applications Programming Interface http://www.cmascenter.org/ioapi) to handle I/O since the initial model inception. In recent years, I/O has been shown as one of the bottlenecks in the CMAQ model system. I/O consumes about 24–35 % of the entire simulation time. For many applications model run time is critically important such air quality forecasting which requires meeting operational time constraints, studies of relationships between climate change and air quality that involve decadal scale simulations (e.g. Gao et al., 2013), or multiscale model simulations involving multiple coarse and fine grid resolutions. For such runtime critical applications improving efficiency of I/O becomes an even more important element that needs to be addressed. To increase the I/O efficiency, we turn to a parallel I/O approach which has been applied in other computer science fields but not for existing environmental models and their processes. Section 2 provides background information about what has been done regarding parallel I/O applications. Section 3 describes the current implementation of I/O in CMAQ through IOAPI. Section 4 depicts our novel technique to enhance I/O performance using pnetCDF and demonstrates I/O enhancement through testing with a smaller scale model. This technique was applied to CMAQ and preliminary results are presented in Sect. 5 while Sect. 6 summarizes the main findings and presents discussion of future work.

2 Previous work in parallel I/O

The independent I/O and collective I/O are the two most common I/O strategies in parallel applications. However, a shortcoming of the independent I/O approach is the servicing of the I/O requests of each process individually (Chen et al., 2010). The collective I/O provides a better solution of managing non-contiguous portions of a file with multiple processes interleaved (Thakur et al., 1999). Several collective I/O techniques are hence developed to improve the parallel I/O performance at various levels by enabling the compute nodes to cooperate with efficient parallel access to the storage system. Examples include, two-phase I/O (del Rosario et al., 1993), data sieving (Thakur et al., 1999), and the collective buffering (Nitzberg and Lo, 1997).

To optimize the I/O performance, software is designed to access non-contiguous patterns by implementation of collective I/O. Data is rearranged and aggregated in memory prior to writing to files, which reduces the number of disk accesses and the seek-time overhead due to large amounts of non-contiguous write requests. Improved I/O efficiency is observed through split writing and hierarchical striping of data (Yu et al., 2007). The benefits of utilizing the improved parallel I/O techniques on applications in various research areas have been recognized (Li et al., 2003; Kordenbrock and Oldfield, 2006; Huang et al., 2014). The approach to parallelize the I/O by using the network Common Data Form (netCDF), a set of software libraries and self-describing, machine-independent data formats, has been discussed regarding the performance of different I/O libraries (Li et al., 2003), including serial netCDF, parallel-netCDF (pnetCDF) and Hierarchical Data Format (Cheng, 2000).

File data striping on parallel file systems also influences I/O performance. Data is distributed using a fixed block size in a round-robin manner among available I/O servers and disks based on a simple striping data distribution function. Optimal striping setup on parallel file systems can significantly reduce the I/O time (Nisar et al., 2012) while inappropriate settings may incur striping overhead for both metadata and file read/write operations (Yu et al., 2007). Research work has shown degradation of parallel I/O efficiency when large numbers of processors are applied to scientific applications such as CMAQ (Kordenbrock and Oldfield, 2006). To overcome these short comings, we re-engineered the current CMAQ I/O module to better utilize more processors on high performance computational machines as well as quantifying the optimal data striping setup on Lustre file systems.

3 I/O in CMAQ

The Community Multiscale Air Quality (CMAQ) modeling system, an active open-source development project of the U.S. Environmental Protection Agency, is an air quality model for regulatory and policy analysis. The interactions of atmospheric chemistry and physics are studied through this three-dimensional Eulerian atmospheric chemistry and transport modeling system. The primary goal for CMAQ is to simulate ozone, particulate matter, toxic airborne pollutants, visibility, and acidic and nutrient pollutant species within the troposphere and across spatial scales ranging from local to hemispheric.

IOAPI, a third-party software, was created concurrently with the initial development of the CMAQ model. It provides a simple interface to handle read and write data in netCDF format in CMAQ. It originally operated in serial mode and was later expanded to run on SMP machines using OpenMP. It has never been implemented with capability to run on a distributed system.

When CMAQ was parallelized in late 1998, a "pseudo" parallel I/O library, PARIO, was created to enable CMAQ to run on a distributed system. PARIO was built on top of the IOAPI library to handle regular data operations (read and write) from each MPI-process. Each individual processor can read its sub-domain portion of data straightly from the input files. However, for output, PARIO requires all processors send its portion of data to the designated processor, i.e. processor 0, which will stitch all data together and write en masse to the output file (Fig. 1a). Clearly, there are a few shortcomings of this strategy: (1) as the number of processors increases, the network will be flooded

GMDD
7, 7427–7459, 2014

with more MPI messages and require longer synchronization time to accomplish an output task, (2) if the domain size remains the same but the number of processors increases, the output data size in each processor decreases which will lower the I/O efficiency, and (3) it requires extra memory for processor 0 to hold the entire dataset before writing to the file.

4 A novel approach

Besides the shortcomings mentioned in Sect. 3, IOAPI has another major drawback, which is not taking advantage of existing technology advancements such as parallel file systems and parallel I/O framework, for example, pnetCDF. Kordenbrock and Oldfield (2006) have shown an enhancement of model I/O performance with the adoption of pnetCDF. Our new approach not only utilizes advanced parallel I/O technology, it also addresses all the shortcomings directly. This new approach performs I/O through pnetCDF on a parallel file system basis, thus eliminating the first and third shortcomings discussed above.

Spatial domain decomposition is widely used in parallelizing scientific models such as CMAQ. The key characteristic of this new technique is data aggregation which can be considered as mitigation for the second shortcoming described above. Generally speaking, data can be either aggregated along the row dimension or column dimension of a rectangular horizontal grid to enhance the I/O efficiency. During aggregation, a small number of localized MPI communication processes were introduced which does not diminish the overall benefit of the technique.

In order to determine the performance of this new technique, a small-scale code was devised. This smaller code, which is designed to mimic the CMAQ model, contains a time step loop with artificial workload. Data is output at the end of each time step. This small scaled code was tested with three time steps and was run on two different machines. The following two subsections provide brief information about the machines as well as how the test was setup.

GMDD
7, 7427–7459, 2014

4.1 High-Performance Computational Systems (HPCs)

The experiments were performed on two HPCs to examine the CMAQ I/O performance with various methods. (1) Edison: a Cray XC30 system with 236 Tflop s^{-1}, 5200 computes nodes with 64 GB memory per node, 333 TB of aggregate memory, Cray Aries high-speed interconnect and 6.4 PB of scratch storage space. Each node has dual-socket twelve-core Intel Ivy Bridge processors at 2.4 GHz. (2) Kraken: a Cray XT5 system with the peak performance of 1.17 Petaflops s^{-1}, 112 896 compute cores, 147 TB of compute memory, 3.3 PB of raw parallel file system disk storage space, and 9408 compute nodes. Each node has two 2.6 GHz six-core AMD Opteron processors (Istanbul), 16 GB of memory, and is connected by Cray SeaStar2+ router.

The file system on both HPCs is managed by Lustre, a massively parallel-distributed file system that has the ability to distribute the segments of a single file across multiple object storage targets (OSTs). A striping technique is applied when a file with a linear sequence of bytes is separated into small chunks. Through this technique, the bandwidth of accessing the file and the available disk space for storing the file both increase as read and write operations can access multiple OSTs concurrently. The default value of stripe count is 1 OST of stripe count and 1 MB of stripe size on both Kraken and Edison.

4.2 Experimental design

To examine the I/O performance of each module, a small scaled model (pseudo code I/O module) written in Fortran90 which includes the basic functions, reading data, writing data and performing arithmetic in between read and write operations, was tested to imitate the complex CMAQ model with the emphasis on the I/O behavior. The code loops for three times to represent three time steps as in regular CMAQ simulations. Three domain sizes were chosen to represent the typical 12 km resolution settings in the CMAQ community: a small domain that covers the entire State of California and vicinity area (CA), $89 \times 134 \times 35 \times 146$ (column by row by layer by species),

GMDD
7, 7427–7459, 2014

a medium-sized domain that covers the Eastern US (EUS) $279 \times 299 \times 35 \times 146$, and a large domain that covers the entire continental US (CONUS), $459 \times 299 \times 35 \times 146$ (Fig. 2). Various combinations of stripe counts (2, 5, 11, 20, and 40) and stripe sizes (1 MB, 2 MB, 4 MB, 8 MB and 16 MB) are tested on different processor configurations (4×4, 4×8, 8×8, 8×16, and 16×16 on CA and EUS domain and 4×8, 8×8, 8×16, 16×16, and 16×32 on CONUS domain). Regular domain decomposition is applied on the spatial domain (column by row) as in the CMAQ model. Each experiment was carried out multiple times and the averaged values were reported. Four different I/O techniques were setup: the serial I/O scheme used in the current CMAQ model which uses regular Network Common Data Form (rnetCDF), I/O implementation with straight parallel netCDF (pnetCDF) (Fig. 1b), our new technique with data row-wise aggregation among MPI processes plus I/O through using pnetCDF (pnetCDFcr) (Fig. 1c), and our new technique with data column-wise aggregation among MPI processes plus I/O through using pnetCDF (pnetCDFcc) (Fig. 1d). Figure 1 illustrates the essence of these methods. Timing includes the actual I/O time (disk writing time) plus any additional time such as data gathering as in the method shown in Fig. 1a or additional MPI communication as needed in data aggregation techniques.

The results provided by the small scaled model serve as a basis to determine the optimal striping information (count and size) for further experiments with the pre-released CMAQ version 5.0.2. One-day simulations of CMAQ on a 4 km resolution EUS domain ($423 \times 594 \times 14 \times 146$) were run to evaluate the differences among rnetCDF, pnetCDF, and the data aggregation schemes using pnetCDF with respect to I/O performance. These tests were conducted on Kraken and Edison with three different processor configurations: 10×12, 12×15, and 18×20.

GMDD
7, 7427–7459, 2014

5 Results

5.1 Small scale model results

In Figs. 3–8 and 10, a relative performance calculation shown in Eq. (1) is plotted against the stripe counts and sizes:

$$\text{rel.performance}\,(\%) = \frac{(T_{\text{m1}} - T_{\text{m2}})}{T_{\text{m1}}} 100\,\% \tag{1}$$

where T_{m1} and T_{m2} denote the averaged maximum I/O time for method 1 and method 2, respectively. Since, all the runs were done on non-dedicated system environments, the same setup was run multiple times for consistency purposes and outliers were filtered. To avoid visual blocking when displaying negative values below the xy plane, absolute values are plotted and solid bars represent positive values and light colored bars represent negative values. In each figure, a larger positive solid bar is the desirable outcome.

The CA case, which represents a relatively small domain, shows a general trend that performance degrades as the stripe count increases and/or the stripe size increases. For this case, pnetCDF performance can be worse than the serial approach using regular netCDF (Fig. 4). With the data aggregation technique, aggregation along the row dimension is better. Overall, data aggregation along row dimension, pnetCDFcr outperforms pnetCDF. Setting the stripe count to 5 and stripe size to 1 MB seems to be the "optimal" settings on both machines and among all processor configurations. Furthermore, as the number of processors increase, the relative performance of pnetCDF drops from ~ 50 to ~ 20 % on Kraken (Fig. 3) and from ~ 40 % to about negative 25 % on Edison (Fig. 4). Conversely, with the optimal settings, relative performance of pnetCDFcr increases as the number of processors increases increases from ~ 20 to 75 % except 4 × 4 case on Kraken (Fig. 3) and increases from ~ 20 to 80 % on Edison (Fig. 4).

GMDD
7, 7427–7459, 2014

The EUS case, which represents a moderately large domain, shows similar result as in the CA case. Relative performance of aggregation along row dimension is much better than along column dimension (Figs. 5 and 6). With small number of processors scenarios, 4×4 and 4×8, pnetCDF performs better than pnetCDFcr. At 8×8, pnetCDFcr performs better than pnetCDF slightly, ∼ 10 %. As the number of processors grows, the enhancement becomes more significant. Overall, the optimal setting on Kraken is stripe count equals to 11 and stripe size equals to 2 MB and on Edison is stripe count equals to 5 and stripe size equals to 2 MB. Again, with pnetCDF, the relative performance drops as the number of processors increases (decreases from ∼ 90 to ∼ 75 % on Kraken and ∼ 50 to ∼ 40 % on Edison). However, the pnetCDFcr shows the opposite behavior: as the number of processors increases, the relative performance increases significantly.

The CONUS case represents a relatively large domain, showing similar results as the CA and EUS cases (Figs. 7 and 8). When the number of processors increases, the relative performance of pnetCDF decreases (from ∼ 80 down to ∼ 60 % on Kraken and from ∼ 75 down to ∼ 50 % on Edison). However, the relative performance of the pnetCDFcr scheme increases dramatically. Overall the "optimal" settings are stripe count equals to 11 and stripe size equals to 2 MB.

5.2 Stripe size and stripe counts effect with PnetCDF

Stripe size and stripe count are two of the key factors that affect I/O performance as shown in Figs. 3–8. The CONUS domain is chosen with various stripe counts (2, 5, and 11) and stripe sizes (1, 2, 4, 8 and 16 MB) here to summarize these effects (Fig. 9). Among all stripe counts, the cases using stripe counts of 11 demonstrate the best performance compared to other stripe counts; for stripe sizes, the 2MB cases were better than the other stripe sizes. As more processors were applied, larger stripe sizes resulted in decreasing performance in writing out data while 2MB cases had relatively better results compared to the other four sizes. Shorter writing time was found when fewer processors were requested. The stripe count effect showed that stripe counts of 11 had the best performance compared to the other two stripe count cases. The differences became more significant when more processors were used.

5.3 The impact of large number of processors on I/O efficiency

Section 5.1 has shown pnetCDF performance decreases as the number of processors increases. When the number of processors continues to increase, the performance of pnetCDF reaches a point that's worse than the serial I/O scheme (Fig. 10). In contrast, pnetCDFcr scheme continues to improve significantly as the number of processors increases.

The I/O efficiency is defined as the rate of data being output. In parallel applications with a spatial domain decomposition strategy, the domain size in each processor becomes smaller as the number of processors increase (Fig. 11, left panel). It is known that the I/O rate is higher if a large chunk of data is being output. Figure 11 (right panel) reflects this assertion which was tested on Kraken. When the data is aggregated, no matter whether it is along row or column dimension, it will increase the data size in the processor which is responsible for the I/O. This is clearly shown in Fig. 11 left panel. With data aggregation (pnetCDFcc or pnetCDFcr), the data size decreases slower than the pnetCDF approach as the number of processors increases. This translates into a higher I/O rate in aggregated schemes than the pnetCDF approach with respect to the same number of processors. pnetCDFcc is worse than pnetCDFcr due to the internal data alignment in the netCDF internal format (row major).

5.4 Application to CMAQ

Based on this small-scale code experiment, the setting of 11 stripe count and 2MB stripe size is selected to employ in a real CMAQ application: one-day simulation on a 4 km resolution EUS domain ($423 \times 594 \times 14 \times 142$). Figure 12 shows the overall writing time recorded on Kraken and Edison with respect to three different ways to perform I/O: the current way using regular netCDF (rnetCDF), using straight pnetCDF with 11 stripe count and 2 MB stripe size (pnetCDF), and our new approach (pnetCDFcr) with 11 stripe counts and 2 MB stripe size. Clearly pnetCDFcr shortens the writing time significantly.

6 Conclusions

We performed a series of experiments with four different I/O modules to examine their I/O efficiencies in CMAQ. First, a small scaled code was tested on three different domains: CA, EUS and CONUS which represents small, moderately large, and large data sizes of CMAQ outputs. The I/O modules include serial mode which is currently used in CMAQ, direct application of parallel netCDF (pnetCDF), and a new technique based on data aggregation which can be along row or column dimension (pnetCDFcr and pnetCDFcc) before applying the parallel netCDF technique. The experiment results show: (1) pnetCDFcr performs better than pnetCDFcc, (2) pnetCDF performance deteriorates as the number of processors increases and becomes worse than serial mode when certain large numbers of processors are used, (3) even though pnetCDFcr does not perform as well as pnetCDF in the small number of processors scenarios, it does out-perform pnetCDF once the number of processors becomes larger. In addition, an overall "optimal" setting has been shown based on the experiments: 5 stripe count and 1 MB stripe size for small domain, 11 stripe count and 2 MB stripe size or 5 stripe count and 2 MB stripe size for the moderately large domain, and 11 stripe count and 2 MB stripe size for the large domain.

This data aggregation I/O module was also tested for a one-day, 4 by 4 km EUS domain using CMAQ compared to the serial I/O mode, which is currently implemented in CMAQ, and direct parallel netCDF. The results show significant reduction of I/O writing time when this new data aggregated pnetCDF (pnetCDFcr) technique is used compared with serial I/O approach and with application of straight pnetCDF. With this finding, the overall run time of scientific applications which require I/O will be significantly reduced. A more important implication is that it allows users to use a large number of processors to run applications and still maintain a reasonable parallel speedup thereby deferring speedup degradation governed by the Amdahl's Law. Furthermore, the technique can be transfered to other environmental models that have large I/O burdens.

Acknowledgements. This paper has been subjected to EPA review and approved for publication. Yang Gao was partly supported by the Office of Science of the U.S. Department of Energy as part of the Regional and Global Climate Modeling Program. The Pacific Northwest National Laboratory is operated for DOE by Battelle Memorial Institute (DE-AC05-76RL01830). The Kraken is a supercomputing facility through National Science Foundation TeraGrid resources provided by National Institute for Computational Sciences (NICS) under grant numbers TG-ATM110009 and UT-TENN0006.

Disclaimer. The views expressed here are those of the authors and do not necessarily reflect the views and policies of the U.S. Environmental Protection Agency (EPA) or any other organization participating in this research.

References

Byun, D. W. and Schere, K. L.: Review of the governing equations, computational algorithms, and other components of the Models-3 Community Multiscale Air Quality (CMAQ) modeling system, Appl. Mech. Rev., 59, 51–77, 2006.

Chen, Y., Sun, X.-H., Thakur, R., Song, H., and Jin, H.: Improving parallel I/O performance with data layout awareness, Cluster '10, in: Proceedings of the IEEE International Conference on Cluster Computing 2010, Washington, DC, USA, IEEE Computer Society, 2010.

Cheng, A. and Folk, M.: HDF5: High performance science data solution for the new millennium, in: Proceedings of SC2000: High Performance Networking and Computing, ACM Press and IEEE Computer Society Press, Dallas, TX, 2000.

del Rosario, J., Brodawekar, R., and Choudhary, A.: Improved parallel I/O via a two-phase run-time access strategy, Workshop on I/O in Parallel Computer Systems at IPPS '93, April 1993, 56–70, 1993.

Fu, J. S., Dong, X., Gao, Y., Wong, D., and Lam, Y. F.: Sensitivity and linearity analysis of ozone in East Asia: the effects of domestic emission and intercontinental transport, J. Air Waste Manage., 62, 1102–1114, 2012.

GMDD

7, 7427–7459, 2014

Gao, Y., Fu, J. S., Drake, J. B., Lamarque, J.-F., and Liu, Y.: The impact of emission and climate change on ozone in the United States under representative concentration pathways (RCPs), Atmos. Chem. Phys., 13, 9607–9621, doi:10.5194/acp-13-9607-2013, 2013.
Huang, X. M., Wang, W. C., Fu, H. H., Yang, G. W., Wang, B., and Zhang, C.: A fast input/output library for high-resolution climate models, Geosci. Model Dev., 7, 93–103, doi:10.5194/gmd-7-93-2014, 2014.
Kordenbrock, T. H. and Oldfield, R. A.: Parallel I/O Advancements in Air Quality Modeling Systems, Poster on 5th annual CMAS conference, Chapel Hill, NC, USA, 16–18 October, 2006.
Li, J., Liao, W.-K., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B., and Zingale, M.: Parallel netCDF: A High-Performance Scientific I/O Interface, in Proceedings of Supercomputing: The International Conference for High Performance Computing, Networking, Storage and Analysis, SC 03, IEEE Computer Society Press, Phoenix, AZ, p. 39, November 2003.
Nisar, A., Liao, W.-K., and Choudhary, A.: Delegation-based I/O mechanism for high performance computing systems, IEEE T. Parall. Distr., 23, 271–279, doi:10.1109/TPDS.2011.166, 2012.
Nitzberg, B. and Lo, V.: Collective buffering: Improving parallel I/O performance, Proceedings of the Sixth IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society Press, Portland, OR, 5–8 August, 148–157, 1997.
Otte, T. L., Pouliot, G., Pleim, J. E., Young, J. O., Schere, K. L., Wong, D. C., Lee, P. C. S., Tsidulko, M., McQueen, J. T., Davidson, P., Mathur, R., Chuang, H.-Y., DiMego, G., and Seaman, N. L.: Linking the Eta Model with the Community Multiscale Air Quality (CMAQ) modeling system to build a national air quality forecasting system, Weather Forecast., 20, 367–384, 2005.
Thakur, R., Gropp, W., and Lusk, E.: Data sieving and collective I/O in ROMIO, in: Proceedings of the Seventh Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, 182–189, 1999.
Wong, D. C., Pleim, J., Mathur, R., Binkowski, F., Otte, T., Gilliam, R., Pouliot, G., Xiu, A., Young, J. O., and Kang, D.: WRF-CMAQ two-way coupled system with aerosol feedback: software development and preliminary results, Geosci. Model Dev., 5, 299–312, doi:10.5194/gmd-5-299-2012, 2012.

Yu, W., Vetter, J., Canon, R. S., and Jiang, S.: Exploiting lustre file joining for effective collective IO, in: The Seventh IEEE International Symposium on Cluster Computing and the Grid, 2007.

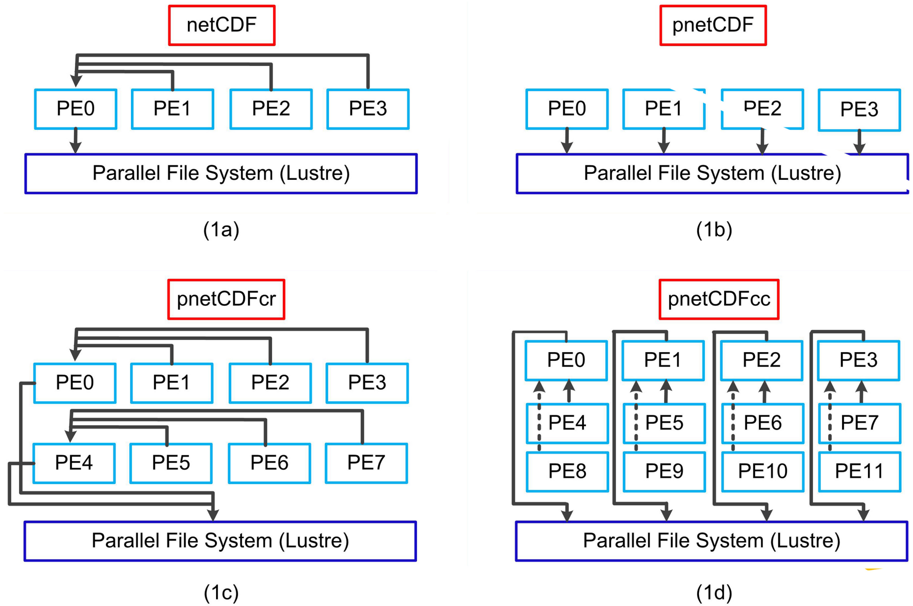

Figure 1. Conceptual diagrams of four I/O modules: serial I/O used in current CMAQ with netCDF data format **(a)**, straight pnetCDF implementation **(b)**, with data aggregation along row dimension and then use pnetCDF **(c)**, and with data aggregation along column dimension and then use pnetCDF **(d)**. PE denotes a processor. Arrows show the direction of data movement.

GMDD
7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

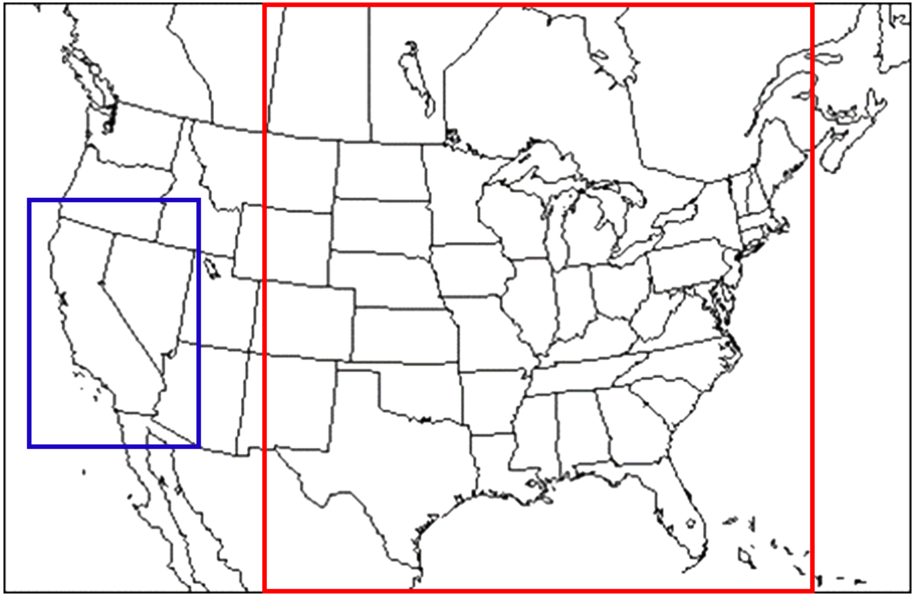

Figure 2. Regional representation of the CA (blue box), EUS (red box) and CONUS (entire) domains.

GMDD
7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

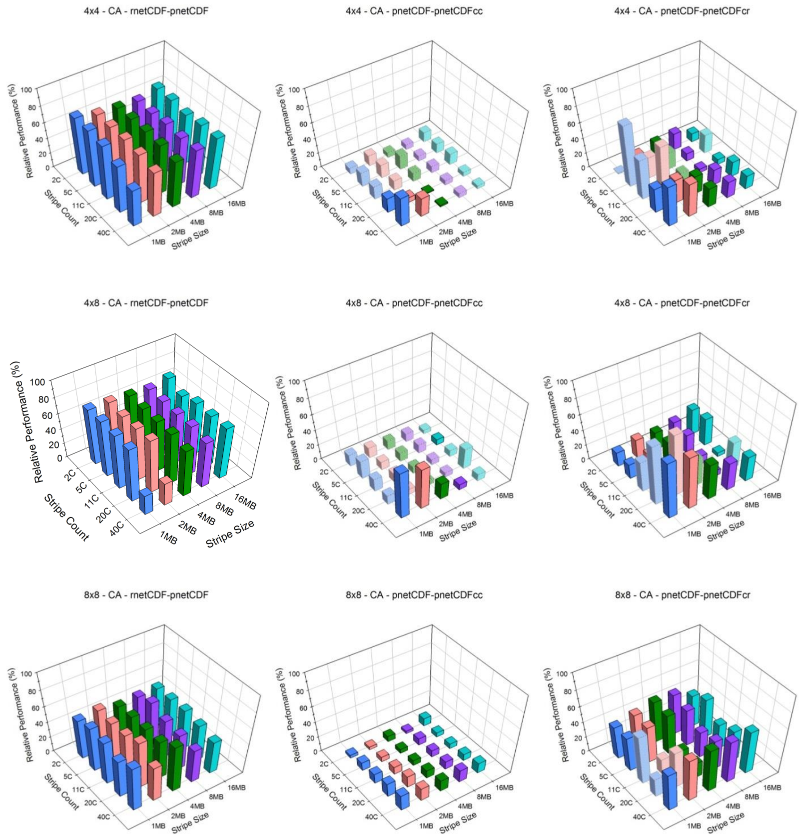

Figure 3. Relative I/O performance of pnetCDFcr and pnetCDFcc to pnetCDF, and pnetCDF to rnetCDF on CA domain with 4×4, 4×8, 8×8, 8×16, and 16×16 processor configuration from Kraken. Same colors represent applying the same stripe size. Solid bar denotes positive value in relative performance while light colored bar denotes negative value in relative performance.

GMDD
7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

Title Page
Abstract | Introduction
Conclusions | References
Tables | Figures
◀I | ▶I
◀ | ▶
Back | Close
Full Screen / Esc
Printer-friendly Version
Interactive Discussion

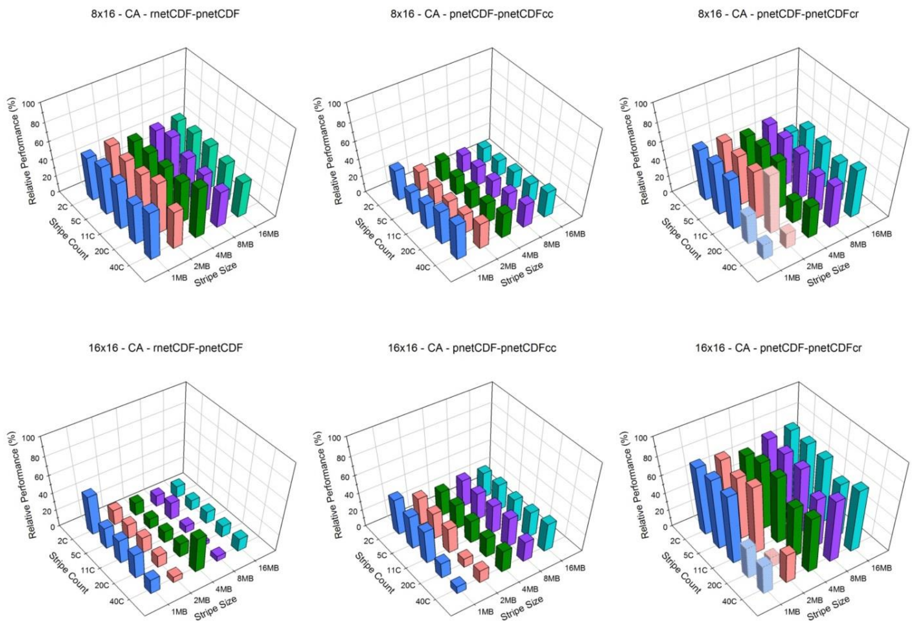

Figure 3. Continued.

GMDD
7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

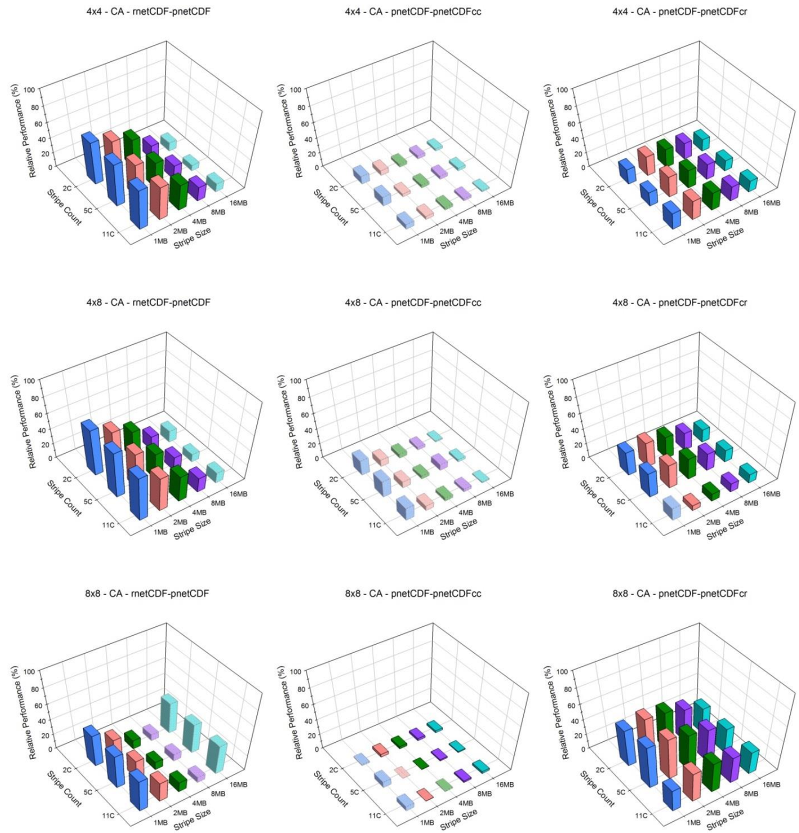

Figure 4. Relative I/O performance of pnetCDFcr and pnetCDFcc to pnetCDF, and pnetCDF to rnetCDF on CA domain with 4×4, 4×8, 8×8, 8×16, and 16×16 processor configuration from Edison. Same colors represent applying the same stripe size. Solid bar denotes positive value in relative performance while light colored bar denotes negative value in relative performance.

GMDD
7, 7427–7459, 2014

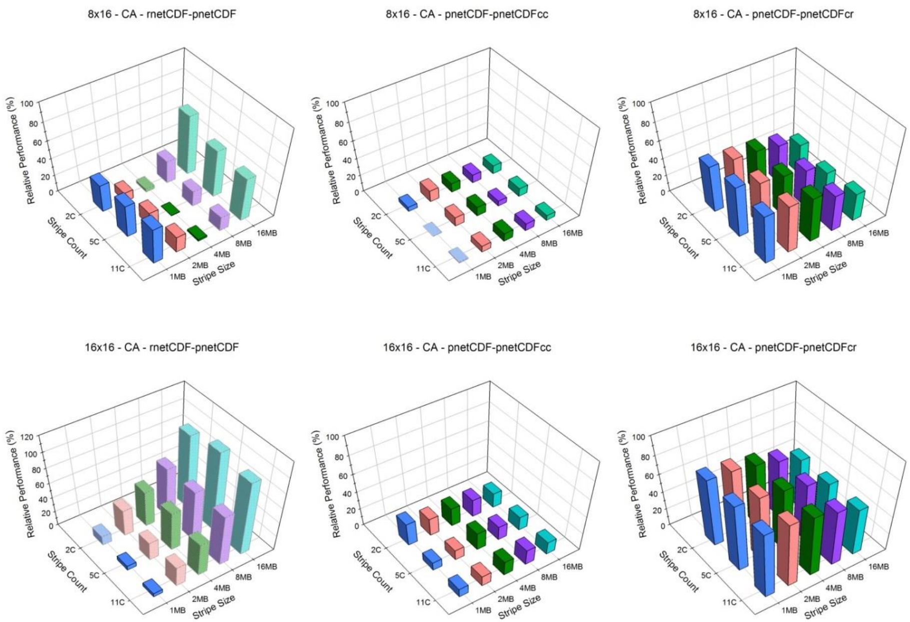

Figure 4. Continued.

GMDD
7, 7427–7459, 2014

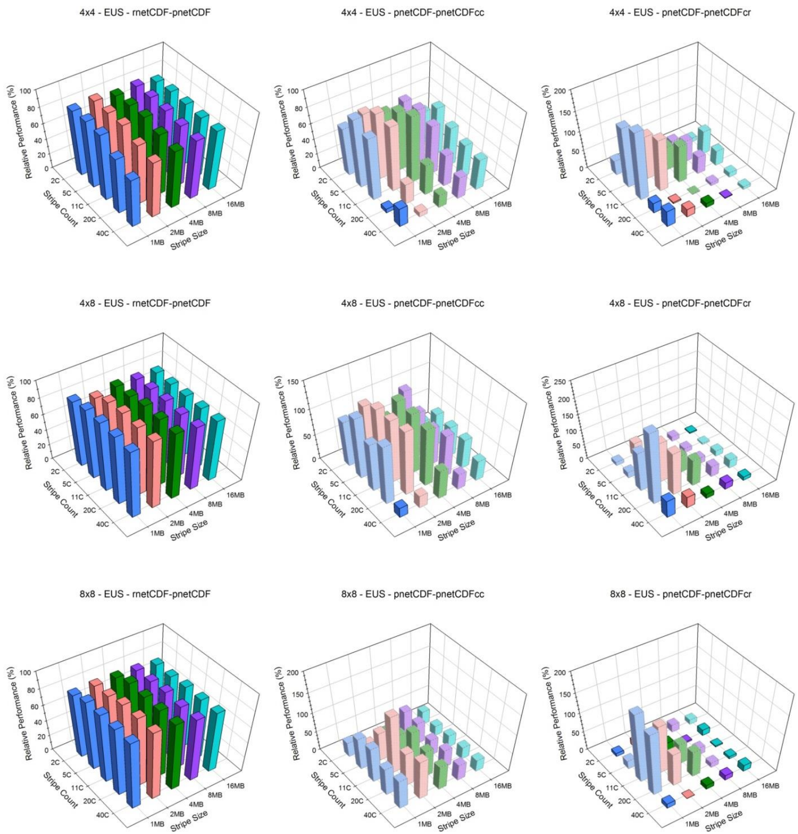

Figure 5. Relative I/O performance of pnetCDFcr and pnetCDFcc to pnetCDF, and pnetCDF to rnetCDF on EUS domain with 4×4, 4×8, 8×8, 8×16, and 16×16 processor configuration from Kraken. Same colors represent applying the same stripe size. Solid bar denotes positive value in relative performance while light colored bar denotes negative value in relative performance.

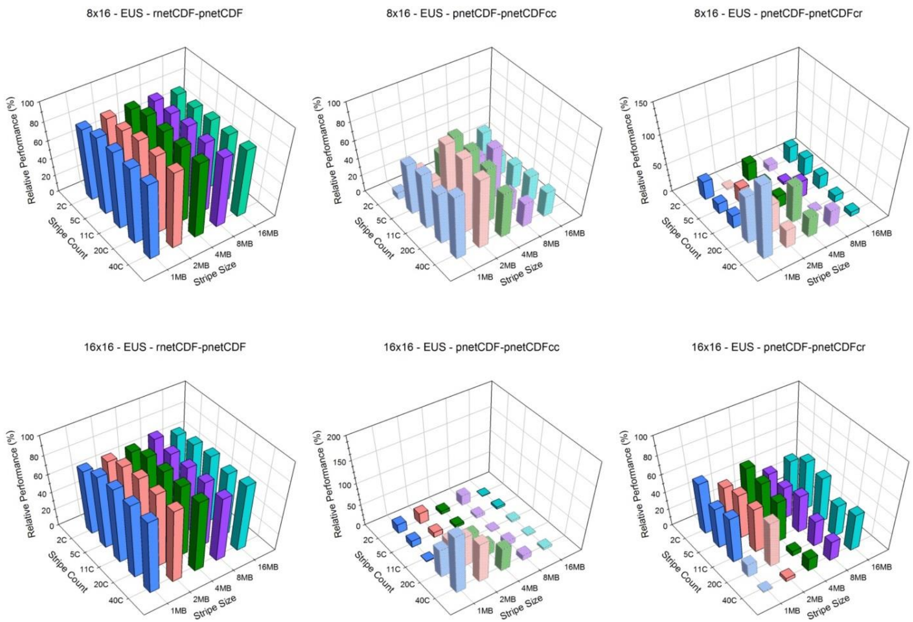

Figure 5. Continued.

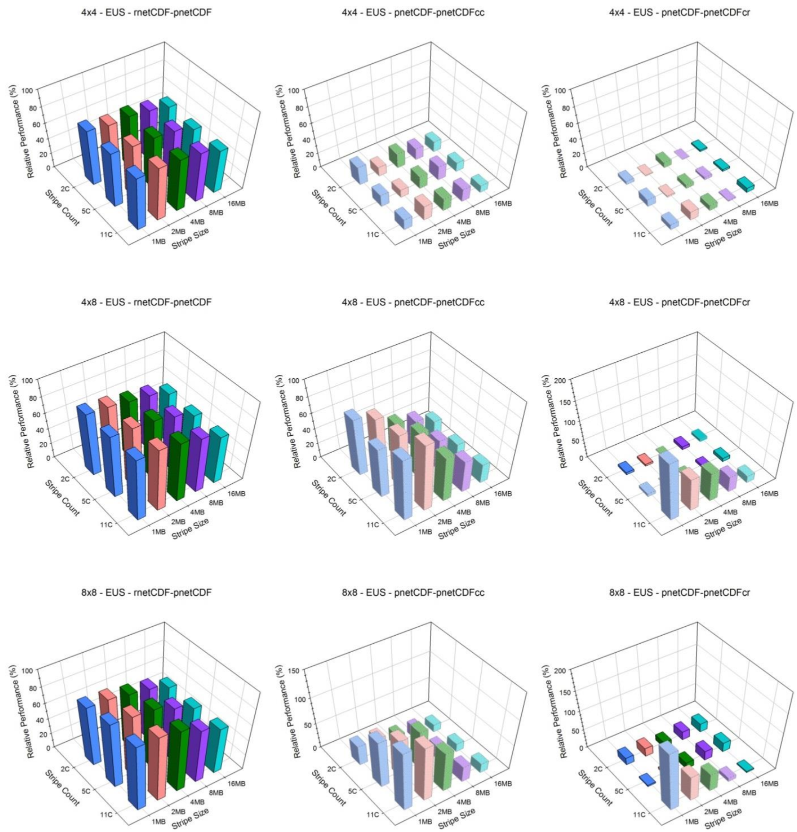

Figure 6. Relative I/O performance of pnetCDFcr and pnetCDFcc to pnetCDF, and pnetCDF to rnetCDF on EUS domain with 4×4, 4×8, 8×8, 8×16, and 16×16 processor configuration from Edison. Same colors represent applying the same stripe size. Solid bar denotes positive value in relative performance while light colored bar denotes negative value in relative performance.

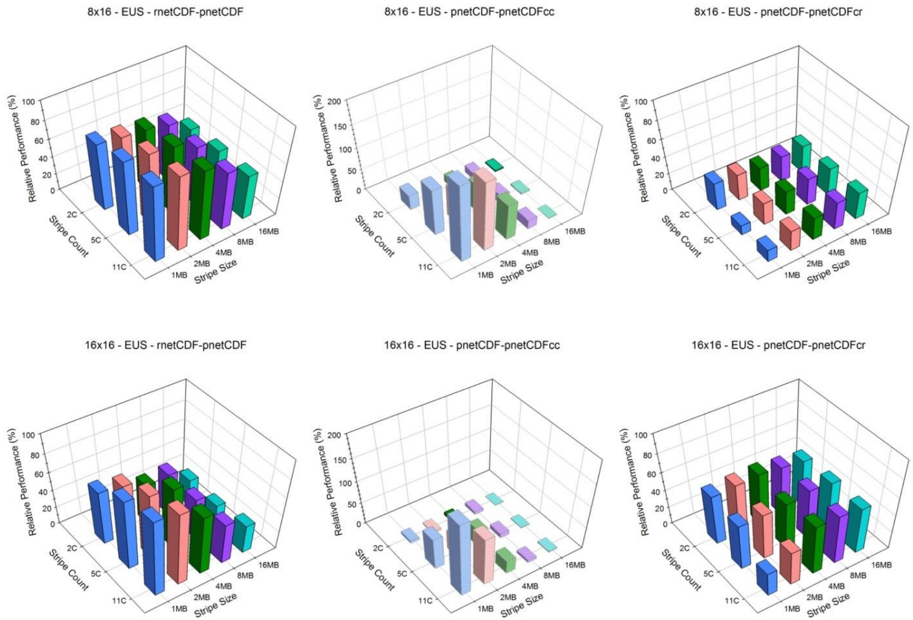

Figure 6. Continued.

GMDD
7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

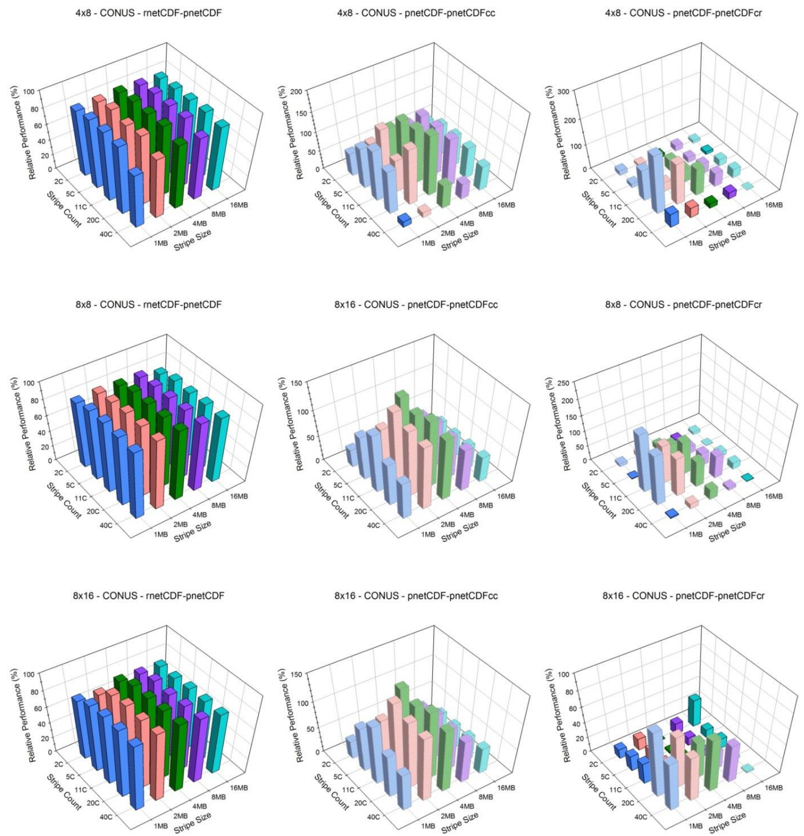

Figure 7. Relative I/O performance of pnetCDFcr and pnetCDFcc to pnetCDF, and pnetCDF to rnetCDF on CONUS domain with 4×8, 8×8, 8×16, 16×16, and 16×32 processor configuration from Kraken. Same colors represent applying the same stripe size. Solid bar denotes positive value in relative performance while light colored bar denotes negative value in relative performance.

GMDD
7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

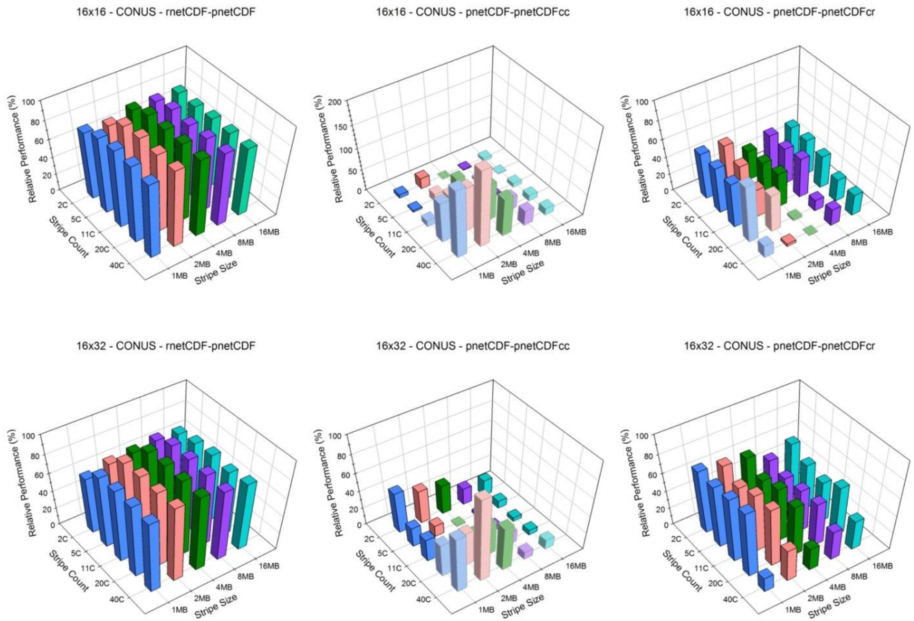

Figure 7. Continued.

GMDD
7, 7427–7459, 2014

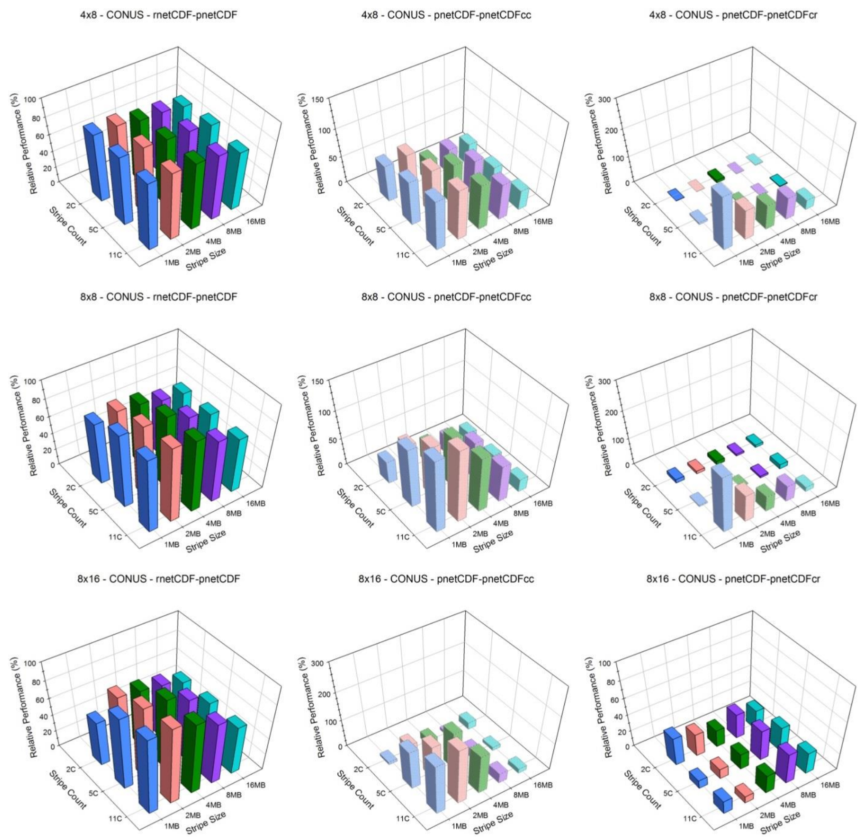

Figure 8. Relative I/O performance of pnetCDFcr and pnetCDFcc to pnetCDF, and pnetCDF to rnetCDF on CONUS domain with 4 × 8, 8 × 8, 8 × 16, 16 × 16, and 16 × 32 processor configuration from Edison. Same colors represent applying the same stripe size. Solid bar denotes positive value in relative performance while light colored bar denotes negative value in relative performance.

GMDD
7, 7427–7459, 2014

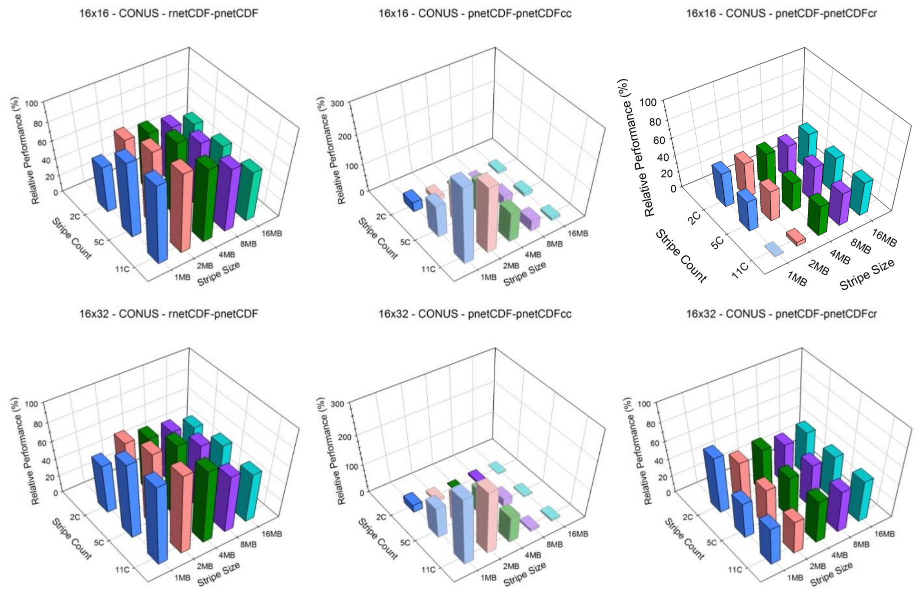

Figure 8. Continued.

Discussion Paper | Discussion Paper | Discussion Paper | Discussion Paper |

GMDD

7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

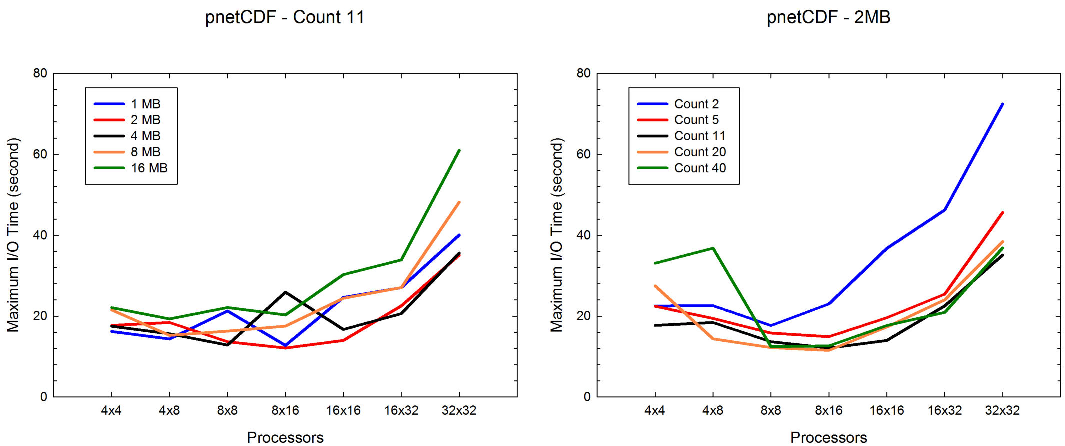

Figure 9. The impact of stripe count and size on parallel netCDF I/O performance on CONUS domain. Left: various stripe sizes with fixed 11 stripe counts. Right: various stripe counts with fixed 2 MB stripe size.

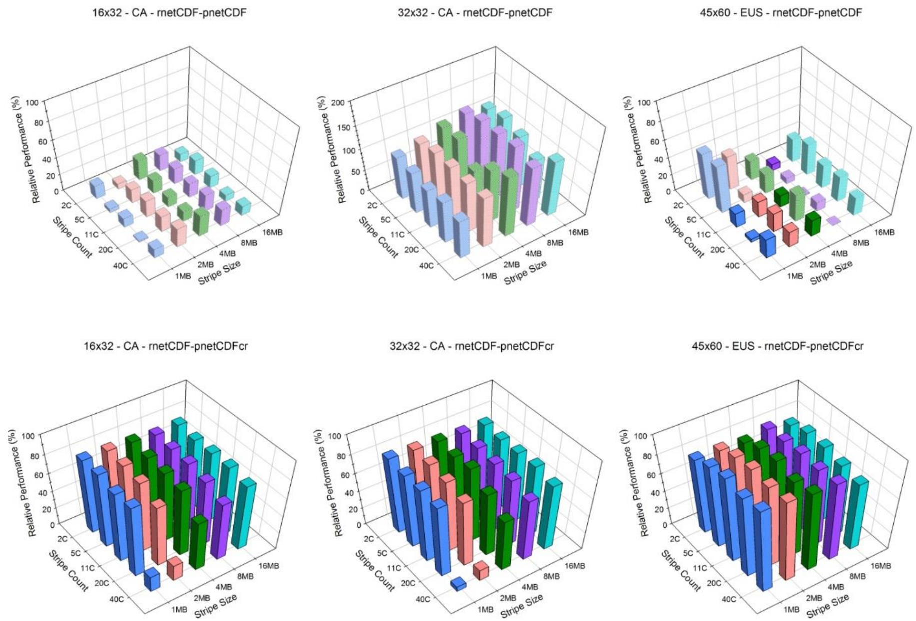

Figure 10. Relative performance of pnetCDF and pnetCDFcr with respect to rnetCDF in a large number of processors scenario on Kraken.

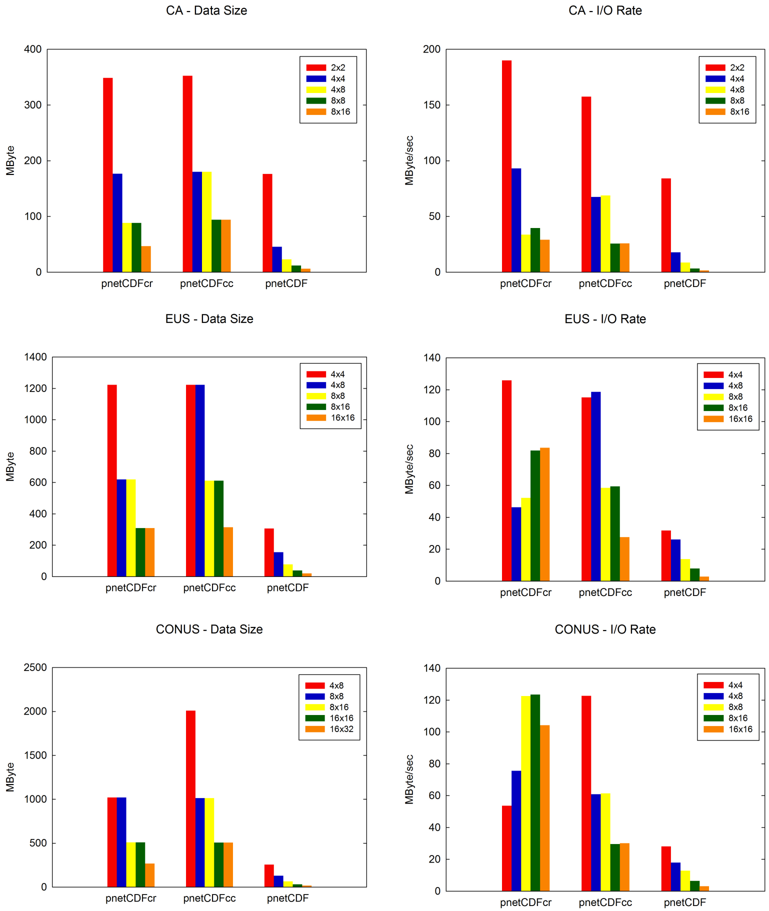

Figure 11. The maximum data size (left panel) and I/O rate (right panel) among all I/O processors in CA (top), EUS (middle) and CONUS domain (bottom), respectively, in the pseudo code experiment running on Edison.

GMDD
7, 7427–7459, 2014

An approach to enhance pnetCDF performance in environmental models

D. C. Wong et al.

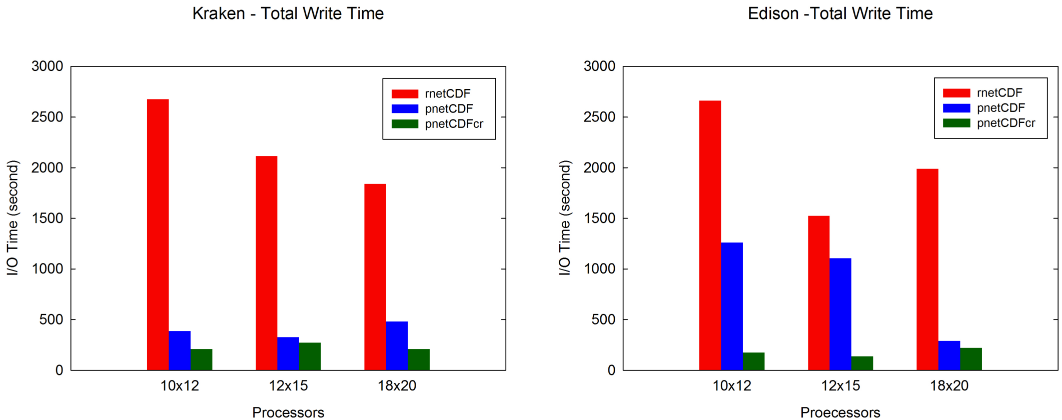

Figure 12. Total write time in one-day CMAQ simulation by different I/O approaches on a 4 km EUS domain with stripe size 2 MB and stripe count 11 (Kraken left and Edison right).

GMDD
7, 7427–7459, 2014