Data Organization and I/O in a Parallel Ocean Circulation Model

Chris H.Q. Ding and Yun He
NERSC Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720

Abstract

We describe an efficient and scalable parallel I/O strategy for writing out gigabytes of data generated hourly in the ocean model simulations on massively parallel distributed-memory architectures. Working with Modular Ocean Model, using netCDF file system, and implemented on Cray T3E, the strategy speeds up I/O by a factor of 50 in the sequential case. In parallel case, on 32 processors up to 512 processors, our implementation writes out most model dynamic fields of 969 MB to a single netCDF file in 65 seconds, independent of the number of processors. The remap-and-write parallel strategy resolves the memory limitation problem and requires minimal collective I/O capability of the file system. Several critical optimizations on memory management and file access are carried out, ensuring scalability and speeding up numerical simulation due to the improved memory organizations.

1 Introduction

Modeling ocean circulation and its influence on the global climate is a grand challenge in computational science. The ocean flow dynamics and physical processes involve a broad range of spatial and temporal scales, requiring decade-long integrations at fine resolutions. Recent advances in numerical ocean modeling [1, 2, 3, 4, 6, 7, 8] has greatly increased our understanding of these processes. State-of-art massively parallel supercomputers provide gigabytes of memory and teraflop computations necessary to run these simulations.

There are a number of critical issues regarding ocean model simulations on distributed-memory parallel computers, such as efficiency in the usage of cache-based processors, load balance due to irregular land contours and bottom topography, scalability of the basic numerical algorithms.

In this paper, we focus on another critical issue, the data I/O in the ocean modeling. During the simulations, gigabytes of data are generated hourly, including snapshots of the 2D and 3D velocity fields and tracers. Effective and efficient I/O strategy and implementations to handle this large amount of I/O in MPP environment therefore become critical for the production simulations.

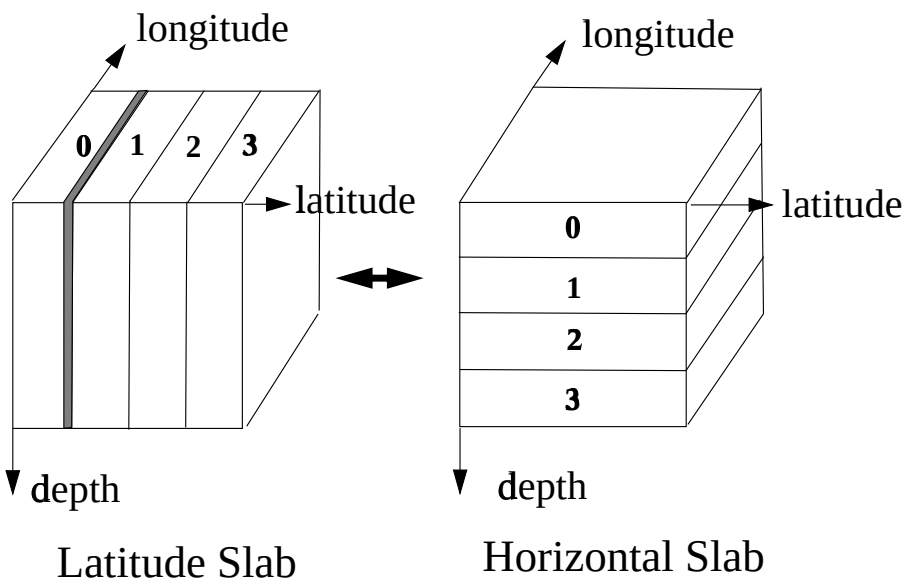

Figure 1: Data layout of the ocean model on 4 processors indicated by 0, 1, 2, 3. Computations are carried out in the latitude-slab decomposition, where latitude slabs are split among total P=4 processors. Note that a latitude slab contains words which are not contiguous in output file space. The horizontal slab decomposition on D designated processors are used for parallel I/O, from this decomposition entire fields on each processor are written out in one shot to a contiguous block in file space. The number D is flexible: D could be anywhere between 1 and the number of vertical layers Nz, depending on memory and I/O channel considerations. Typically D is much smaller than P (although in this Figure we assume $P = D = 4$ for simplicity).

The ocean model we study in this report is the Modular Ocean Model version 3 (MOM3) [4, 5] developed at Geophysical Fluid Dynamics Laboratory (GFDL) which is a three-dimensional general ocean circulation model, widely used in the oceanographic community to simulate ocean and ocean-related events.

2 Model Description

The ocean model solves the primitive equations governing large scale ocean circulation with Boussinesq and hydrostatic approximations [1, 2, 3]. It uses finite difference scheme on the discreted domains. The dynamics split into a barotropic mode, involving depth-averaged column velocities (2D variables), and a baroclinic mode, including deviations from barotropic mode and other 3D tracers. It uses a free surface formulation with an explicit finite difference method.

The codes are written in Fortran, with about 80,000 lines in 350 subroutines. There are a large number of physical parameterization options. Currently it uses a 1-dimensional (latitude dimension, see Figure 1) decomposition for both vectorization on vector architecture and parallelism on MPP with message passing.

3 Memory Organization and Data I/O

A number of codes features make the data I/O in the ocean model complex. The codes use a memory window scheme to slice through the complete data set stored in disk files. This out-of-core computing mode saves memory and allows codes running in memory-limited platforms, such as Cray T90 and workstations. On traditional Cray vector supercomputers, the files are stored on solid state disk which is fast, the out-of-core mode works well. On distributed-memory computers where residence memory is plenty and can hold the entire data set, the memory window scheme is still kept and data are copied back and forth between the small memory window and the disk file which now sits in memory (called ramdisk for this reason). This extensive memory copying poses a problem for writing 3D fields out efficiently and slows down the computation significantly.

Figure 1 shows the data layout for the ocean model. For computational efficiency (vectorization) reasons, data fields stored in computer memory are indexed as U(ix,iz,iy), where ix,iy,iz refer to longitude, latitude, and depth, respectively. The latitude dimension is in outer loop for the convenience of vectorization and parallelization. On the other hand, the 3D fields output files require the arrays to be indexed as U(ix,iy,iz), because most data analysis tools deal with this index order. Furthermore, the out-of-core ramdisk file for implementing memory window has another indexing scheme convenient for slicing through data, but differs from the above two indexing orders. These three different storage orders cause some complexity in data output and also slow down the data transfer rates very significantly.

A further complication is that MOM3 uses netCDF file system [9] for its data I/O. NetCDF is self-describing, portable, and flexible file system. It can read or write a local block from/to the file in a single call even though the data are not contiguous in file space. Its main problem is efficiency, and there is also lack of some important functionality in parallel environment. MOM3 uses a netCDF wrapper which is designed for single processor access.

4 Sequential I/O with netCDF

We start with a critical examination of the snapshot routine which writes out main 3D fields such as flow velocities, tracers, and 2D fields such as surface momentum and temperatures. In the existing codes, for each of the 3D field, the data is written out one latitude slice at a time, as

```
for iy=1, Ny
    write entire 2D slice U(ix,iz,iy) with fixed latitude iy
end do
```

The necessary index change to conform to the U(ix,iy,iz) index order is done inside the netCDF file system. Because each latitude slice is not contiguous in file space (see shaded area in Figure 1), each netCDF write is in fact broken into Nz smaller writes. In fact, entire Nx $\times$ Ny $\times$ Nz words are split into Ny $\times$ Nz writes, each of length Nx words. Thus this mode of I/O is quite inefficient.

The new I/O scheme proceeds as the following: (1) reshuffles the data in computer memory to the correct indexing order; (2) writes all latitudes in one shot. This new scheme eliminates the large overhead associated with repeated disk access and reduces the I/O time dramatically. For the $3^o \times 3^o$ resolution case, the writing time is reduced from 54 seconds to 1.1 seconds, a factor of 50 speedup!

This new sequential I/O scheme indicates the overwhelming advantage to re-order the data outside the I/O system calls. Its implication for the parallel I/O is that we should reshuffle the data in memory, using communication network which has communication bandwidth typically 10 times faster than I/O bandwidth. To do this, the entire data fields must reside in memory. That is, an in-core computing mode is necessary.

5 In-Core Computing mode

The purpose of adding an "in-core" mode is two-folds: (a) to achieve a single uniform memory image so that I/O can proceed much more efficiently as discussed before; (b) to increase the computational speed by eliminating redundant data copying between memory window and ramdisk (the out-of-core mode). Use of ramdisk simplifies the inter-processor communication on which the initial GFDL implementation is based upon.

We develop an option in MOM3 to let entire data sets "in-core", i.e., essentially eliminates memory window entirely (in practical implementation, this is accomplished by opening the memory window to the maximum required and eliminating the ramdisk file). Now communication of halo latitude slices is done directly between appropriate dynamic variable arrays on neighboring processors, instead of between ramdisk arrays.

This in-core computing mode speeds up the baroclinic computation by about 40%, due to the elimination of redundant data copies. The communication time is also reduced as a result of improved memory access (the size of communicated data remain unchanged). It also makes other I/O simpler and faster because now one has the entire data array conveniently available as they are updated, instead of storing in the ramdisk in a storage format dictated by the data slicing considerations.

5.1 Optimizations

Several optimizations are performed. One of them is the *getunit*()/*relunit*() which are called in every time step: open a file, write a few diagnostic numbers and close the file. In sequential computing case, these repeated "open/close" file operations do not cause a large overhead. However, in parallel environment, as numerical calculation parts are speeded up by a large factor ($\sim$50 on 64 processors), these repeated file open/close together remains constant time and becomes a significant overhead. For the $3^o \times 3^o$ resolution case, on 16 processors or more, it exceeds all other useful calculation/communication combined together. We corrected this significant overhead by inserting a new "file-open-interval" to keep the file open for many time steps; close and reopen the file at the specified interval. This eliminates the overhead and speeds up this part by a factor of 55!

5.2 Scaling Analysis

Here we use a realistic model problem covers $0 - 360°$E, $77.8°$S - 0N, with $0.5° \times 0.5°$ resolution at the equator using isotropic grid. The model has a grid size of $722 \times 258 \times 40$, using free surface formulation with an explicit solver. Timing for one simulated day (dtts = dtuv = 1800 sec, dtsf = 25 sec) is shown in Figure 2. The in-core version of MOM3 runs about 20% faster than the out-of-core version (initial GFDL implementation). It also scales better (see Figure 3), because of the elimination of memory window and related data transfers. Here we see some speedup points above the ideal curve; this is due to the assumption of the in-core version with 16 processors has a perfect speedup of 16 (the problem size does not fit in less processors).

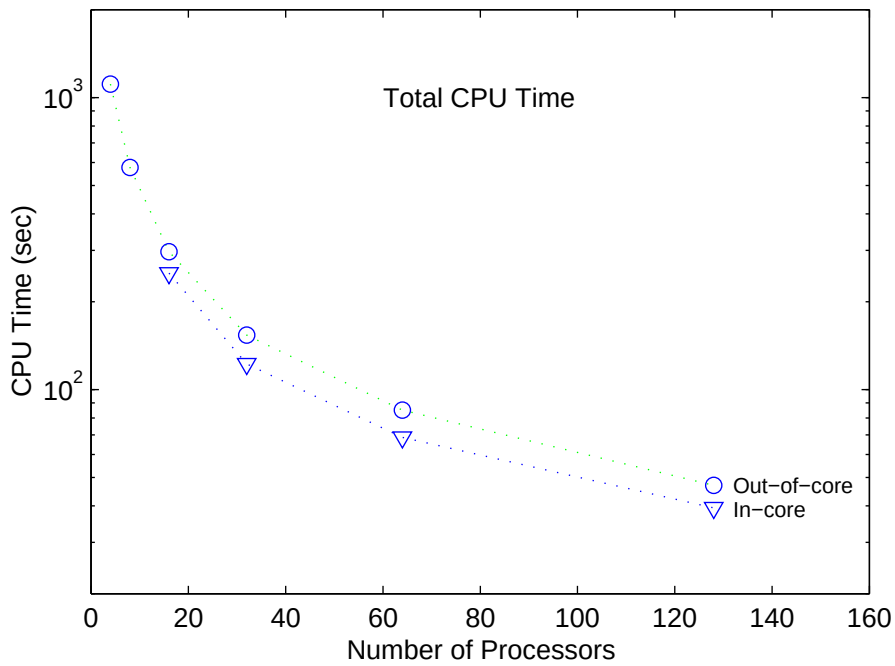

Figure 2: Total CPU time on different number of processors for 0.5^o resolution case with size $722 \times 258 \times 40$, with in-core and out-of-core modes. The in-core mode includes the optimizations described in the text.

Baroclinic equation parts , 3D fields, generally scale up better than the barotropic equation solving (2D fields) as shown in Figure 4. The problem with barotropic scaling is due to the small size messages required in each iteration. In this $722 \times 258 \times 40$ test case, the free surface equation solver time step size is 72 times smaller than the dynamics in baroclinic part, i.e., free surface explicit solver is iterated 72 times per dynamic time step. In a larger scale, fine resolution simulations, this timestep size ratio could increase to more than 100, the barotropic part will dominate the CPU time. The less-than-good scaling of barotropic part will potentially cause severe scaling problem in larger scale simulations.

The times spent on communicating data among processors are shown in Figure 5. Communication times saturate as number of processors increase, as expected for a 1D domain decomposition. The in-core version spends less time in communication because it avoids some of the memory copying.

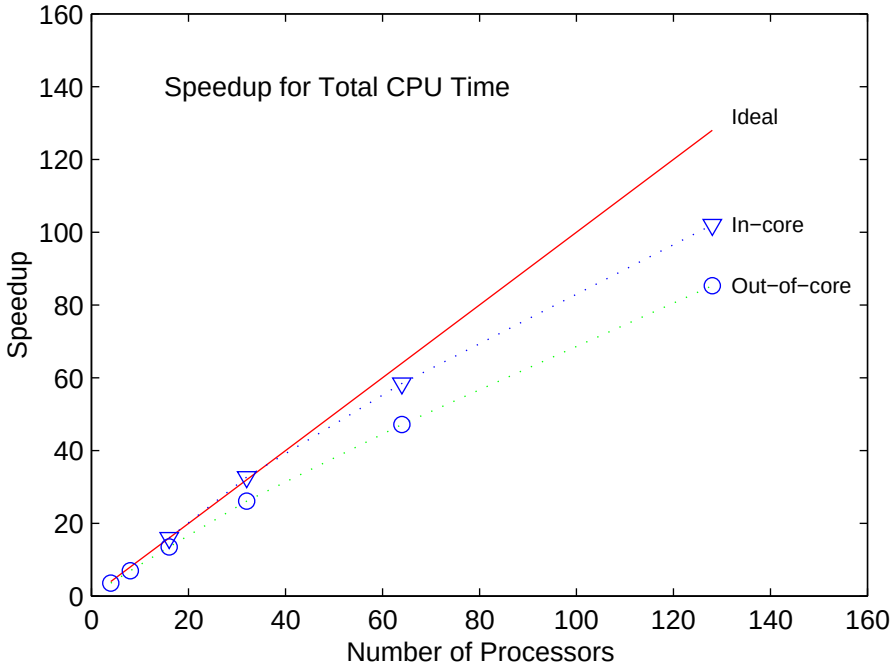

Figure 3: Speedup of total CPU time on different number of processors for 0.5^o resolution case with size $722 \times 258 \times 40$, with in-core and out-of-core modes.

6 Parallel I/O

Our strategy for parallel I/O on MPPs focuses on efficiency and portability, at the expense of extra software structures. This strategy uses a few designated I/O processors. These designated processors reserve buffer area, gather data from other computing processors, reorganize them if necessary, and then write them out as contiguous blocks in parallel. With this approach, data files are written as if they are processed in sequential environment, irrespect of how many processors will access them. This stand-alone file approach has several major benefits:

- It allows the output data files to be directly analyzed and visualized in any other workstation environments, without extra file conversions.
- It makes restart design simple, since the file configuration is independent of the number of processors.
- This in turn makes it adaptable to a changing environment. For example, a model simulation may proceed for some time on 256 processors. Then, after a checkpoint, it can restart on 128 processors.
- It is portable to any other platform, since this approach uses only standard file interface, with the only requirement that multiple processors can write contiguous blocks into the same file in parallel.

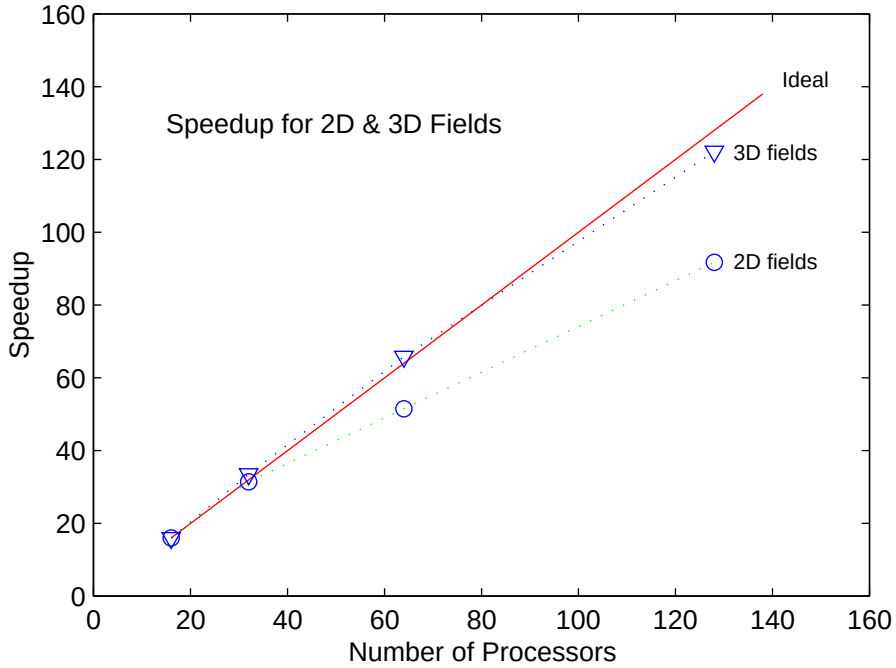

Figure 4: Speedup for 2D and 3D fields using in-core mode for 0.5^o resolution case with size $722 \times 258 \times 40$. 3D fields are prognostic variables such as velocities and tracers, mostly in baroclinic equations; 2D fields are those used in solving barotropic equations. The solid line indicates the ideal speedup, i.e., 100% scaling.

In this approach, we first carry out a global remapping of the 3D distributed array on the P computing processors in the latitudinal slab decomposition (see Figure 1) to the horizontal-slab distribution on the D designated I/O processors (these I/O processors are part of the total P computing processors in the present implementation; they could also be extra dedicated processors for I/O purpose alone, to overlap disk I/O with computation when extremely excessive I/O are required). Once a 3D field is remapped to the designated I/O processors, they can be written out as contiguous blocks in a 1D array, which could be done very efficiently in most file systems on most architectures.

The disadvantages of this approach are:

- Data configuration must fit to the residence memories on these D designated I/O processors. However, this limitation is unlikely to apply to the ocean modeling used in decadal and century-long simulations. In actual implementations, we only need allocate the memory buffer for a single 3D array (which is about 5% of the total required memory), and do I/O for one 3D array at a time, reuse the buffer for all 3D prognostic arrays. Furthermore, we can increase D to reduce buffer size on each designed I/O processors.

- Remapping 3D arrays require additional CPU time and demanding inter-processor communication network. Fortunately, in most MPP architectures, communication bandwidth is typically 10 times higher than I/O bandwidth, and an efficient remapping algorithm has been developed and implemented. So the remapping time is negligible (about 10%) compared to actual disk access time.

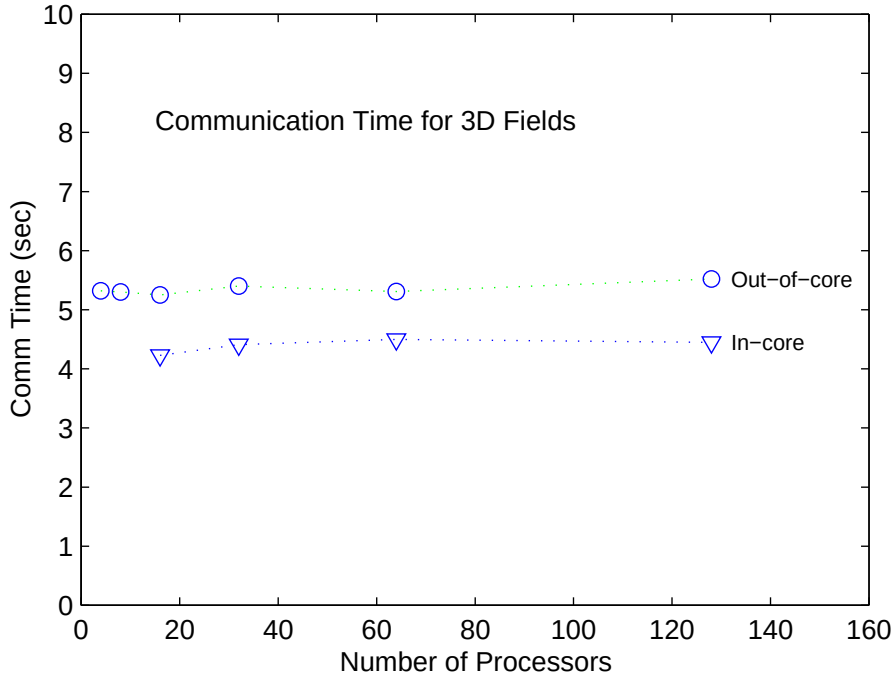

Figure 5: Communication time for 3D fields for in-core mode and out-of-core mode in the 0.5^o resolution case.

Using the in-core computing mode, the data reside in the memory in the latitudinal-slab decomposition (see Figure 1), are most conveniently available for remapping onto designated I/O processors. The I/O module then repeats the following for each prognostic variables, one at a time: (a) remapping the 3D array to the I/O processors, and (b) writing them out from I/O processors using netCDF file system. We discuss these two steps in some detail next and give I/O performance results.

6.1 3D Array Remapping

A stand-alone module for remapping multi-dimensional array on distributed-memory computer is developed [10] for use in the I/O part. It remaps any 3D array in the latitudinal-slab decomposition to the horizontal slab distribution on the D designated I/O processors, to prepare for writing to disk file. The module uses a novel vacancy tracking approach to do in-place local data reshuffle to the correct indexing order, therefore eliminating the need for auxiliary array which must have the same size of the original data set. It combines this with a global block exchange algorithm, leading to an in-place global 3D array remapping module. This generic array remapping software can also be used in other grids-based climate models for polar filtering, spectral transforms, and I/O subsystems.

6.2 Parallel netCDF

When a 3D array is in the horizontal-slab distribution, we write it out to disk file using netCDF file system in parallel, i.e., all the D designated I/O processors write the distributed 3D array to a single file as a 1D distributed array with contiguous blocks, each block on each processor.

A number of important issues of netCDF in parallel T3E environment, including unlimited dimension, use of $NETCDF_FFIOSPEC for I/O control, and opening a global file by subset of processors were resolved by NERSC staff [11]. The parallel netCDF uses the Cray FFIO system "global" layer, a simple collective I/O mode.

When properly setup, all major functionalities of netCDF in sequential environment also work in parallel environment. Different processors can write to different parts in the same file simultaneously, even with data block on different processors where each block goes to discontiguous locations in file space, such as the latitudinal slice shown in Figure 1. The data transfer rate in this case is very low, about 50-100 times slower than the case where data blocks are contiguous in file space, similar to that in the sequential environment we discussed earlier. This is one of the main reasons we decide to remapp the 3D array before writing them out to disk.

Currently, MOM3 uses a netCDF wrapper that interfaces between netCDF and application codes. The wrapper provides many useful functions, but does not allow multiple processors to output file. This forces data on multiple processors to be gathered onto a single processor and writing out from there. This approach has memory limitations and low I/O efficiency. After many experiments with the wrapper, we decide to bypass it in this study.

6.3 Snapshot I/O

Snapshot is the critical I/O part in MOM3. It writes out major dynamic 2D and 3D fields. We implement this part following the remap-write strategy discussed above.

Two model resolutions are investigated. One is 0.25^o resolution with problem sizes $1442 \times 514 \times 40$ and the snapshot output file size 969 MB. Another is 0.5^o resolution with problem size $722 \times 258 \times 40$ and snapshot file size 243 MB. In production computing, these outputs are written out every 1-4 wall clock hours, depending on the timestep and necessary observation frequency.

The results on T3E for snapshot I/O are shown in Figure 6. The computing processors range from 4 to 512, and out of them, $D = 4$ processors are chosen to be the designated I/O processors. $D = 4$ is chosen since it is the minimum number of designated processor required for the model sizes examined in this study. Smaller D would require buffer size that exceeds the available memory (the designated I/O processors also hold all data required by a normal computing processor, in addition to the buffer for I/O purpose).

The total time here includes the file "open", file "close" time, remapping time and writing time, since this represents more realistic timing in production environment. From Figure 5, the total I/O time remains constant with respect to total processors, for both resolutions, indicating good scaling of the I/O strategy. For the 0.25^o resolution with size $1442 \times 514 \times 40$ case, the remapping time reduces

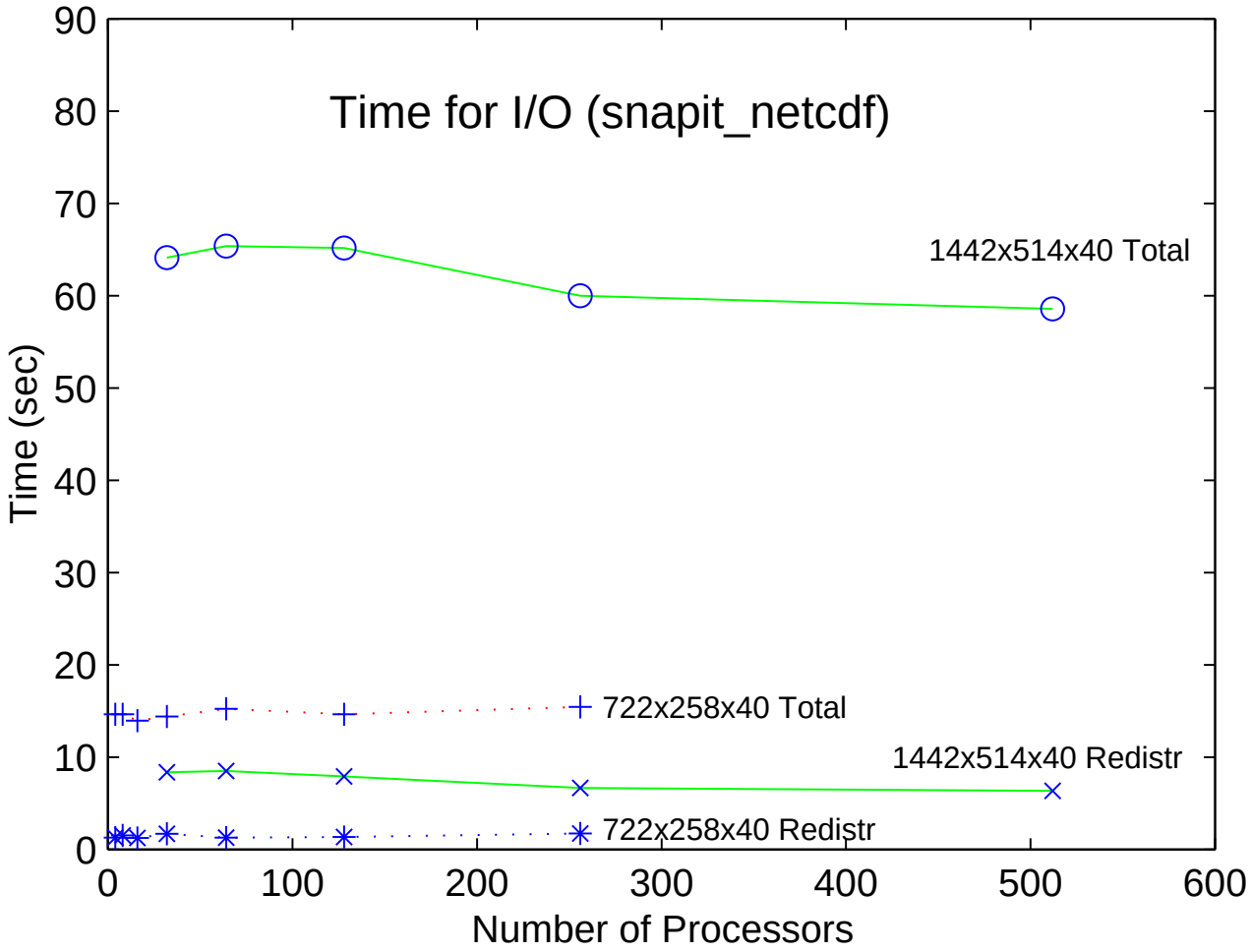

Figure 6: Snapshot I/O timing results for two model sizes, 0.25^o resolution with $1442 \times 514 \times 40$ and 0.5^o resolution with $722 \times 258 \times 40$. Shown are the total I/O time including file "open", "close" and remapping times.

from 8.4 sec on 32 processors to 6.4 sec on 512 processors, showing good scaling of the remapping algorithm. The pure disk access time should remain constant, because they are always done from the four I/O processors, independent of total computing processors. Thus the total I/O time remains about 64 seconds. For the 0.5^o resolution case, snapshot I/O shows very similar characteristics. These test runs indicate that the remap-write strategy is a good scalable I/O approach and it meets the I/O requirement for the size of model problems expected in the coming years.

6.4 MPI-IO

Recently, an implementation of MPI-IO becomes available on T3E. Because MPI-IO does not support netCDF format, we cannot use it directly. Instead, we experimented with MPI-IO on the T3E without netCDF. We found that MPI-IO performs reasonable for simple collective I/O operations such as writing contiguous blocks typical of 1D arrays; however, for the remapping task required in the

ocean model snapshot I/O, MPI-IO performs about 10 times slower than our 3D remap-and-write implementation.

7 Concluding Remarks

In this report, we systematically examine the I/O problem in large scale fine resolution ocean simulation with MOM3. We first identify the difference in data indexing order in memory and in file space, and find the reshuffle-and-write increase the I/O speed by a factor of 50. We then develop an in-core computing option so that data are conveniently available for remapping to the final horizontal-slab distribution for writing out. The in-core mode also speeds up the entire simulation by about 40% because it avoids the data copying between memory window and ramdisk. After many experiments, we develop and implement a remap-and-write I/O strategy that resolves the indexing order difference and overcomes memory limitations. The test runs with two large realistic problem sizes show the total I/O time remains constant from 8 processors up to 512 processors, a very good scaling property. The I/O approach is also portable to many different file systems and architectures because it only requires a file system able to write 1D array collectively to a single file in parallel.

Although this I/O work is designed for MOM3 model, much of the lessons learned and strategies developed could easily be applied to other ocean models, such as the Parallel Ocean Program (POP)[6], Miami Isopycnic Ocean Model (MICOM)[7], and other grids-based climate models. The remapping module and I/O module are available to public upon request (MOM3 codes are available to public at GFDL web site). The modules could be easily modified to accommodating 2D domain decomposition, from the current 1D decomposition in MOM3. For this purpose, the codes are written (in Fortran 90) as a stand-alone library-like module with modular programming techniques.

Acknowledgment. This paper describes MOM3 I/O related work of a collaboration between NERSC and GFDL. R. Pacanowski, V. Balaji, J. Shelton, C. Kerr, S. Griffies, B. Ross at GFDL and R.K.Owen, H. Anand, H. Simon, S. Luzmoor at NERSC are closely involved in the collaboration; without their contribution, this work would not be possible. We thank Ron Pacanowski for valuable discussions on MOM3 codes structures and V. Balaji for T3E implementation related work. We also thank J. Carter for the help on MPI-I/O. This work is supported by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences, and also the Office of Biological and Environmental Research of the U.S. Department of Energy under contract number DE-AC03-76SF00098.

References

[1] K. Bryan. Numerical Methods for the Study of World Ocean Circulation. *J. Comp. Phys.*, 4. 1687-1712, 1969.

[2] M.D. Cox. A Primitive Equation, 3-Dimensional Model of the Ocean. Group Tech Report 1, GFDL, Princeton, NJ, 1984.

[3] A.J. Semtner. Modeling Ocean Circulation. *Science*, 269, 1379-1383, September 1995.

[4] R.C. Pacanowski, R.K. Dixon and A. Rosati. Modular Ocean Model User's Guide. Ocean Group Tech Report 2, GFDL, Princeton, NJ, 1992.

[5] R.C. Pacanowski and S. Griffies. Modular Ocean Model 3 User's Guide. This guide and MOM codes are available from GFDL web site http://www.gfdl.gov/~kd/MOMwebpages/MOMWWW.html

[6] R.D. Smith, J.K. Dukowicz, and R.C. Malone. Parallel ocean general circulation modeling. *Physica*, D60, 38, 1992.

[7] R. Bleck, S. Dean, M. O'Keefe, and A. Sawdey. A Comparison of Data-Parallel and Message Passing Versions of the Miami Isopycnic Ocean Model. *Parallel Computing*, 21, 1695-1720, 1995.

[8] P.G. Eltgroth, J.H. Bolstad, W.P. Dannevik, P.B. Duffy, A.A. Mirin, H. Wang and M.F. Wehner. Coupled Ocean/Atmosphere Modeling on High-Performance Computing Systems. Eighth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, March 1997.

[9] netCDF is a free software developed by UNIDATA. See netCDF web site http://www.unidata.ucar.edu/packages/netcdf/

[10] C.H.Q. Ding. Optimal 3D Array Parallel Remapping on Distributed Memory Architectures LBNL/NERSC Tech Report. April 1999.

[11] For netCDF system development at NERSC, see web site http://www.nersc.gov/~rkowen/netcdf/index.html