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

Chris Ding and Yun He

NERSC Division
Lawrence Berkeley National Laboratory
University of California

Collaboration between NERSC and GFDL

R. Pacanowski, V. Balaji, S. Griffles, J. Shelton, B. Ross
Geophyisical Fluid Dynamics Laboratory

R.K Owen, H. Anand, S. Luzmoor, H. Simon
Lawrence Berkeley National Laboratory

Work funded by U.S. Dept. of Energy
Office Biological and Environmental Research
Office of Computational Technology Research/ MICS

Ocean General Circulation Models

- Modular Ocean Model (MOM)
- Parallel Ocean Model (POP)
- Miami Isopycnic Coordinate Ocean Model (MICOM)
- Semi-spectral Primitive Equation Model (SPEM)
- Several other models

Grand Challenges

- Fine resolution
  - ~ 0.1^o resolution -> 3600x1200x100 grid
- Long time scale (decades, century)
- Many physical processes
- Topography
  - ~ Load balance due to bottom and land

Modular Ocean Model (MOM3)

- Community model, free download from Internet
- Developed and supported by GFDL, Bryan-Cox-Semtner.
- Large number of physical parametrizations.
- Widely used in ocean flow/circulation simulations (350 users)
- Adopted by NCAR (NCOM), LANL(POP), and many others for further developments.

MOM3

- 3D Large scale general ocean circulation
- Navier-Stokes + hydrostatic, Boussinesq approx.
- Finite difference
- Dynamics split into barotropic, baroclinic modes
- Barotropic: depth-averaged column velocities, 2D variables
- Baroclinic: deviations from barotropic modes, 3D variables
- Tracers: temperature and salinity, etc. 3D variables
- Free surface, Killworth et al, explicit method
- Free surface, Dukowicz et al, implicit method.

Some Observations of MOM3 Codes

- Memory window (out-of-core)
- Many options (*.F --> *.f)
- 80,000 lines of Fortran in 367 subroutines.
- I/O uses netCDF
- 1D decomposition (latitudes)

FLow Chart of Simulation Process

Read topography, initial condition, etc

Time step through all subtask:

Load Memory Window data from disk/ramdisk

Advection velocities

Isopycnal mixing, vertical mixing, horizontal mixing

Vertical boundary condition

tracer equations

Baroclinic equations

Write Memory Window data to disk/ramdisk ← **Communication**

Barotropic equation (free surface) ← **Communication**

Diagnostics

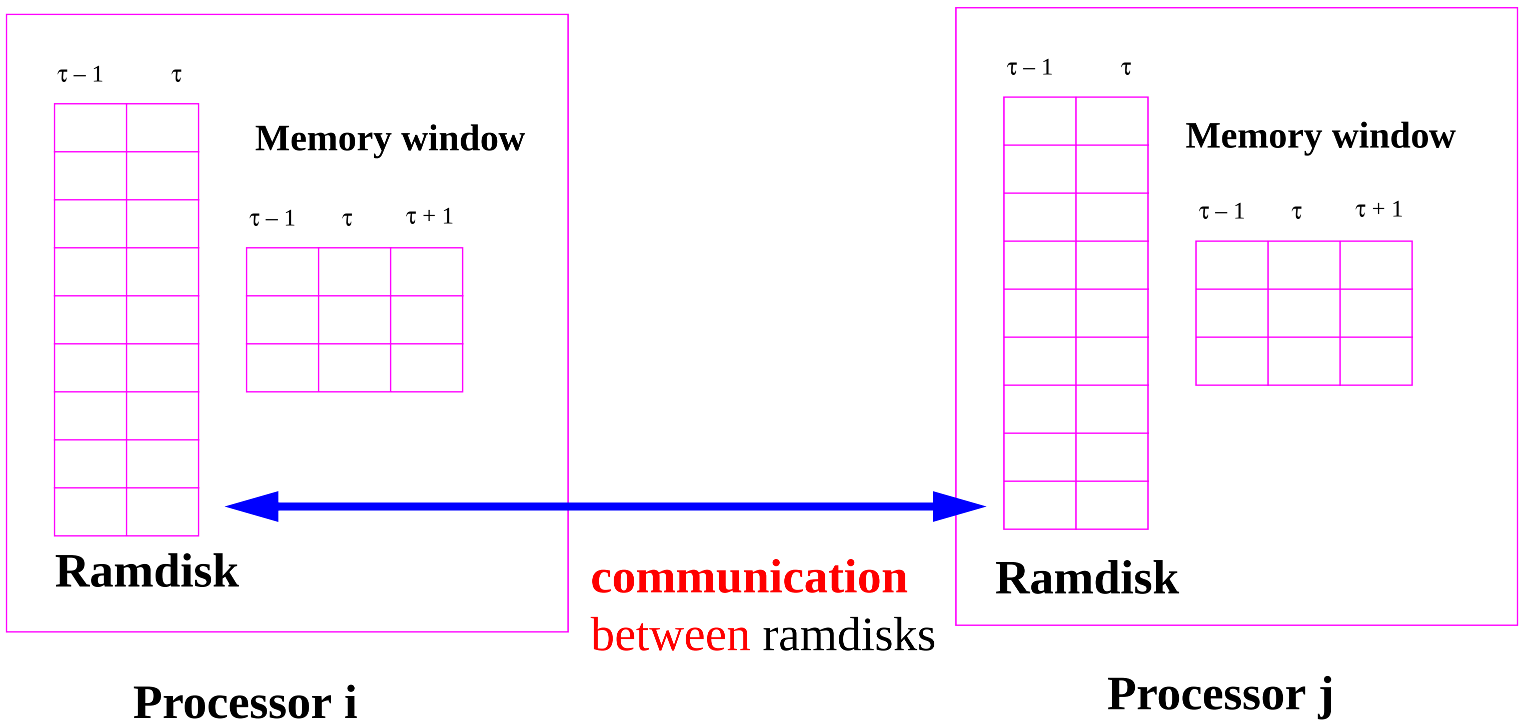
Data Organization

Three different data indexings:

- In memory: A(i , k , j , uv , ts)
- In disk file: A(i , j , k , uv , ts)
- In ramdisk: A(i , k , uv , ts , j)

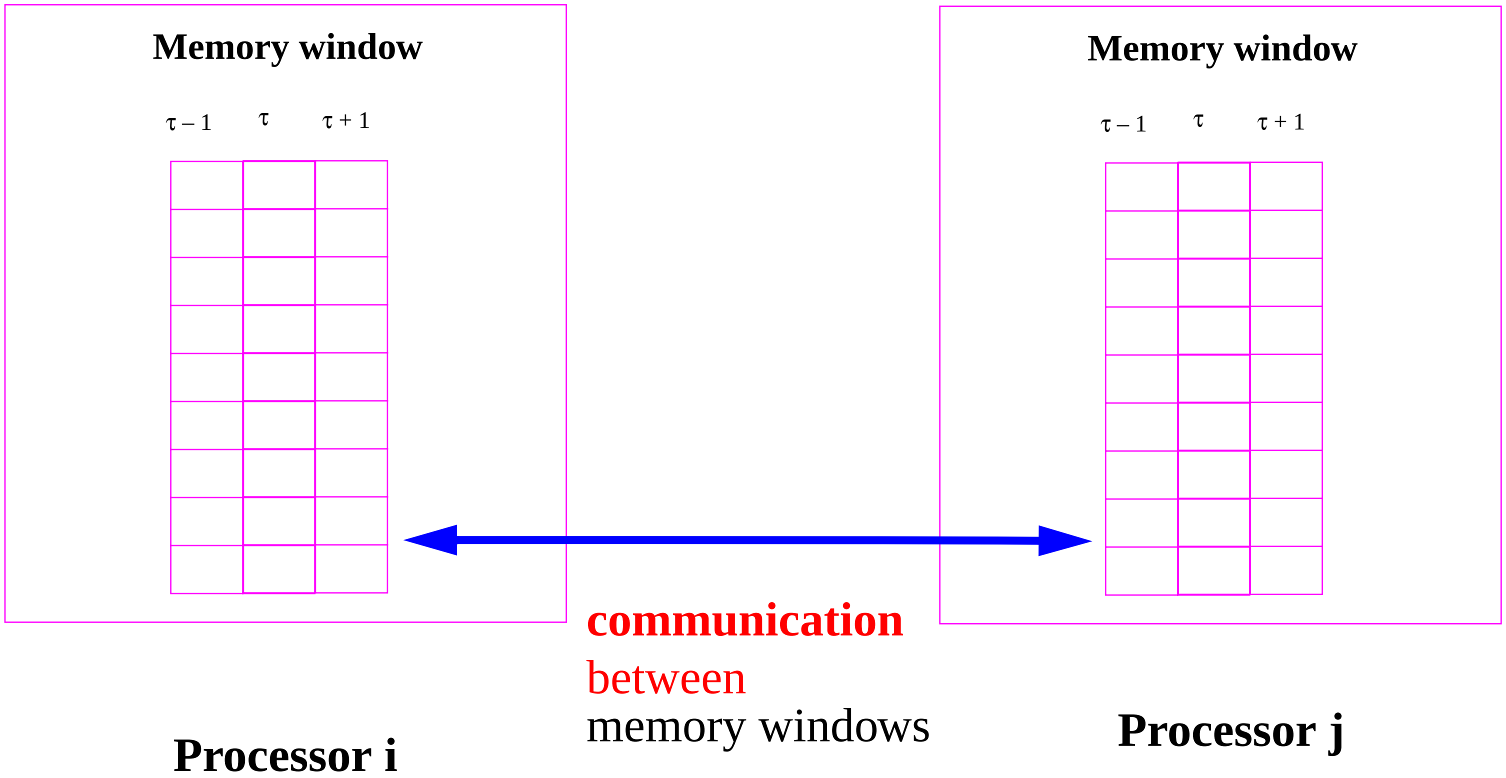
uv=1,2 for velocities
ts=1,2,..., for temp, salinity, etc

Out-of-Core Mode



* Requires extra memory copy

In-Core Mode



* Speedup Computation
* Prepare for parallel I/O

Input/Output, netCDF

MOM uses netCDF (POP, CSM, Impact)

- netCDF is self-describing, portable, flexible.
- main problem is efficiency

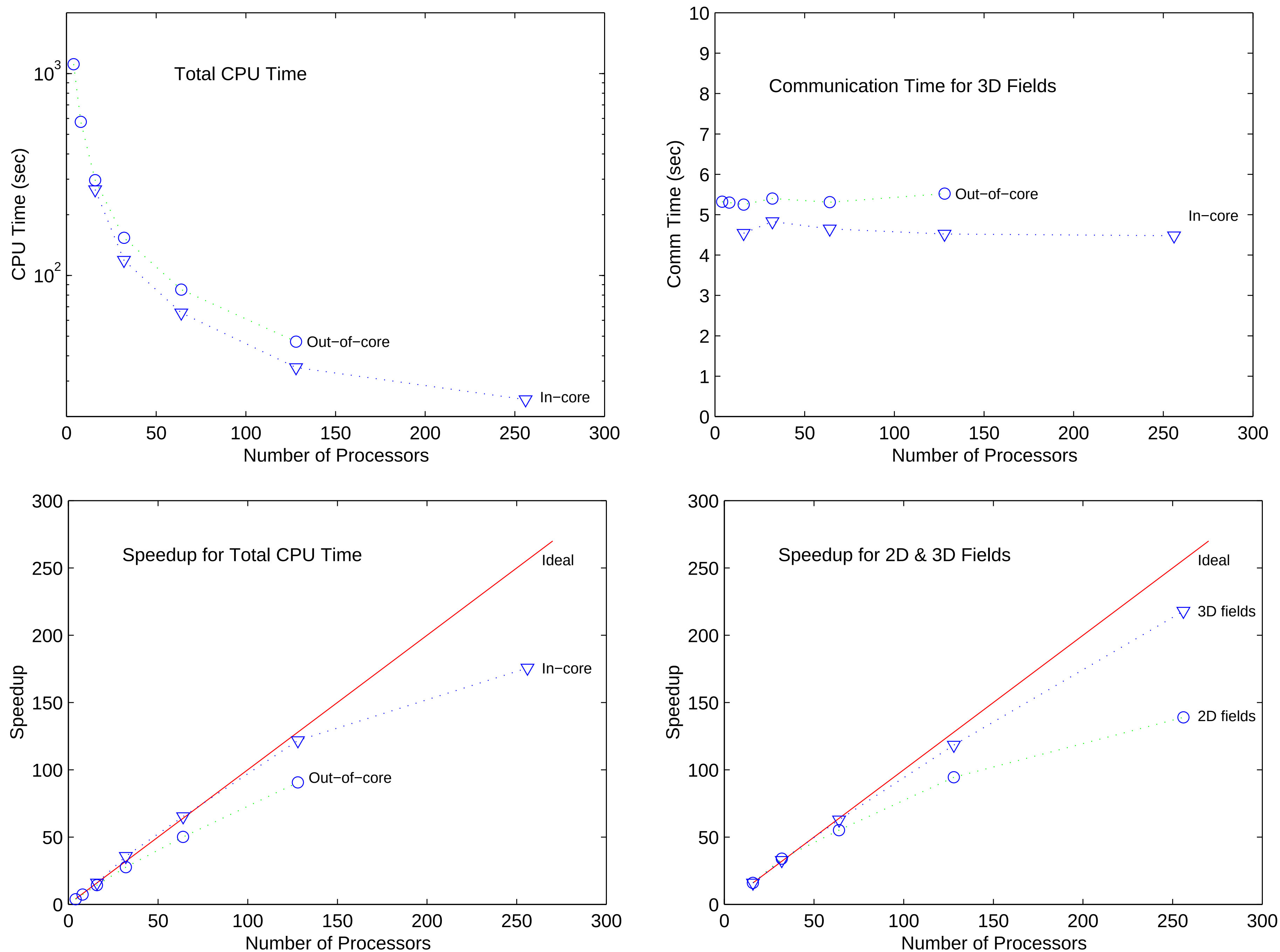
NERSC resolved critical issues of netCDF in parallel T3E environment (Owen, Anand, Luzmoor, Davis)

- Unlimited dimension
- **Assign** I/O control environment ($NETCDF_FFIOSPEC)
- subset of PEs open a global file

netCDF rates are reasonable

- 17 MByte/sec on 1PE.

0.5^o x 0.5^o Global Ocean (722x258x40)



Snapshot I/O in MOM

Old:

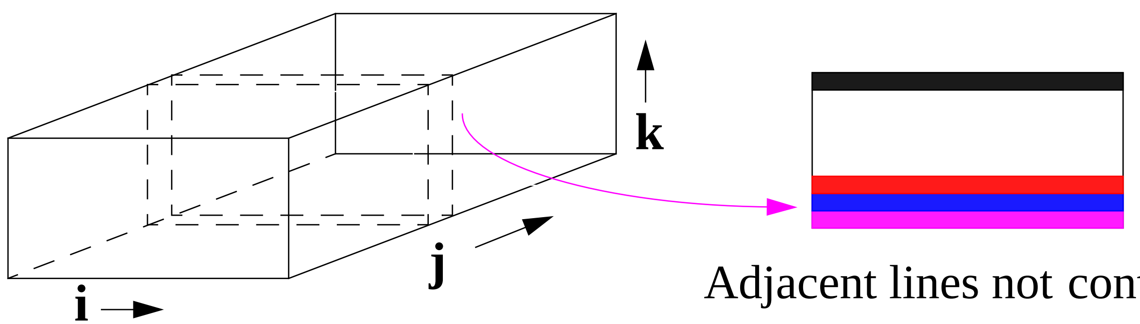
- Load from Ramdrive to Memory Window
- Write out data one latitude slice at a time.

New:

- Do index switch in memory.
- Write all latitudes in one shot.

Reduce I/O time by a factor of 50 !

A(i, k, j)
in memory



A(i, j, k)
in data files

Adjacent lines not contiguous in file space

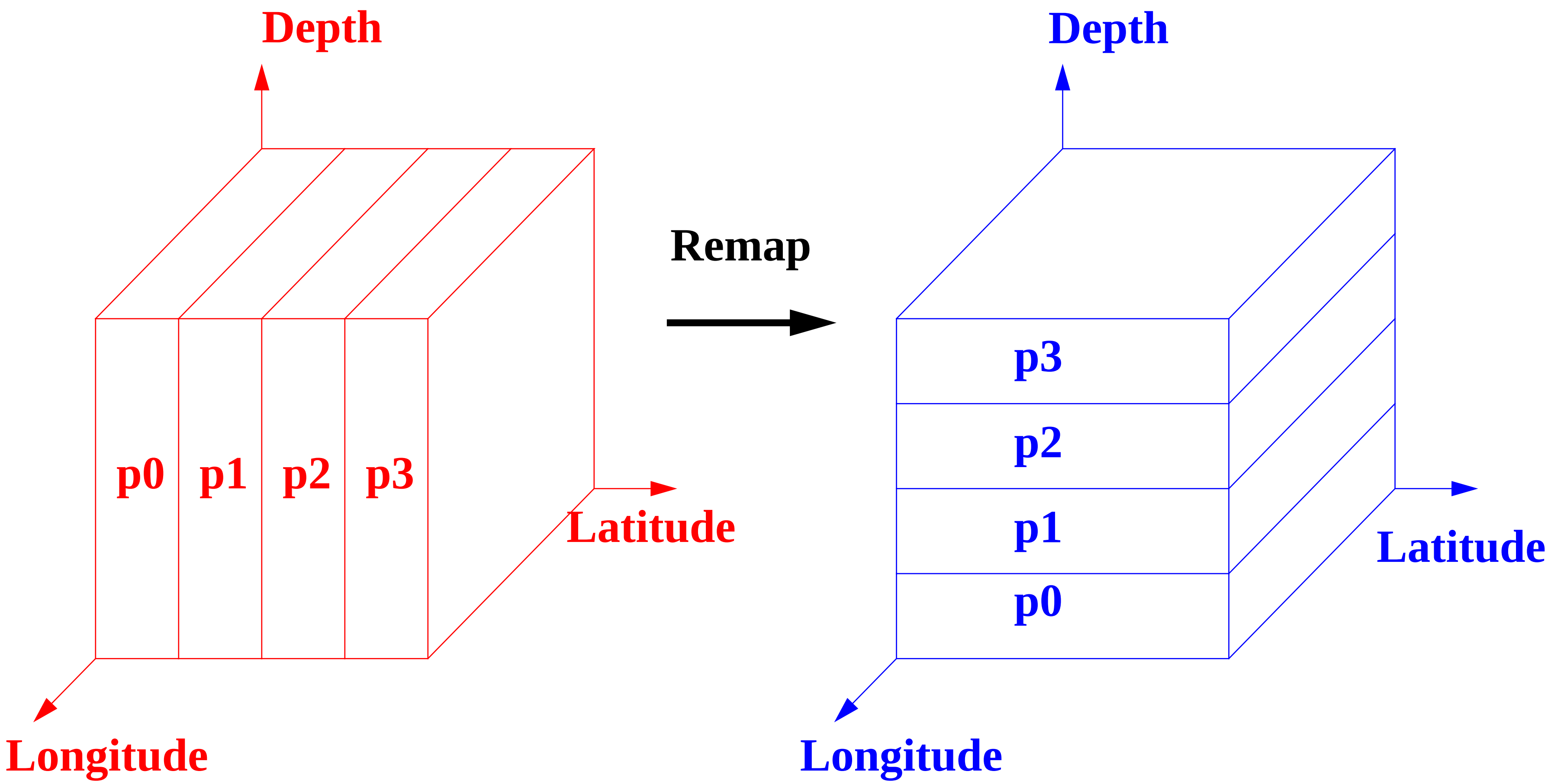
Memory Window (Out-of-core)

- Only small portions of data stored in computer memory. Swap data between memory and disk.
- Enable MOM to run on from workstations to C90
- Complicates code design, parallelization
- 50% slowdown on cache-based processors

Table 1: Timing for Different Memory Usage Modes

Time	Total	Baroclinic	Barotropic
Memory Window Only	256	202	35.3
MW + Ramdrive	354	297	35.4

Remapping 3D Array for Efficient I/O



Latitude-Split
A(i, k, j)

Depth-Split
A(i, j, k)

Parallel I/O Basic Design

- Data file written as in sequential environments
  - ~ Portable to other platforms
  - ~ Output file directly used in any other platforms (visualization) without extra file convertion
  - ~ Restart/checkpointing simplified
  - ~ Adaptable to changing environment: run on different # of processors at different times.

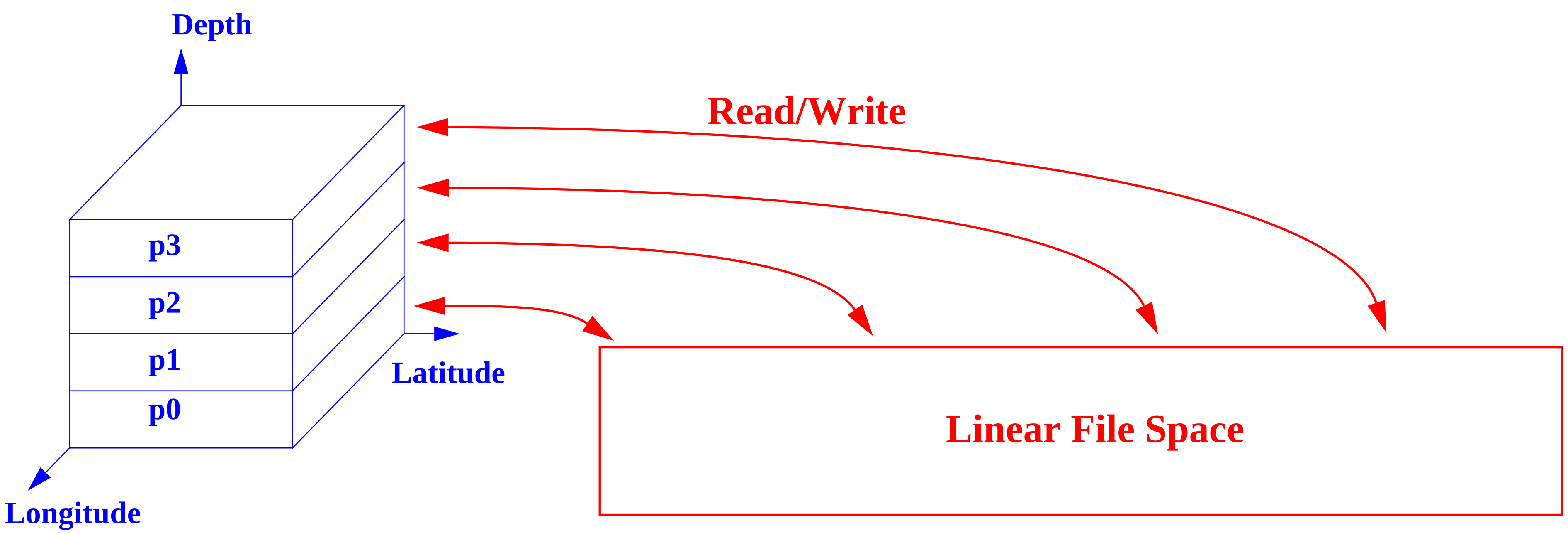
- Use a few designated I/O processors
  - ~ Relieves memory limitations of a single processor
  - ~ Increasing available I/O channels, thus bandwidth

Parallel I/O Basic Design

- Data file written as in sequential environments
  - ~ Portable to other platforms
  - ~ Output files directly analyzed/visualized in any other platforms without extra file convertion
  - ~ Restart/checkpointing simplified
  - ~ Adaptable to changing environment: run on different # of processors at different times.
- Low-level I/O modules
  - ~ Most existing I/O interface intact

Parallel I/O Basic Design (con't)

- Use several designated I/O processors
  - Relieves memory limitations of a single processor
  - Increases available I/O channels, thus I/O rates

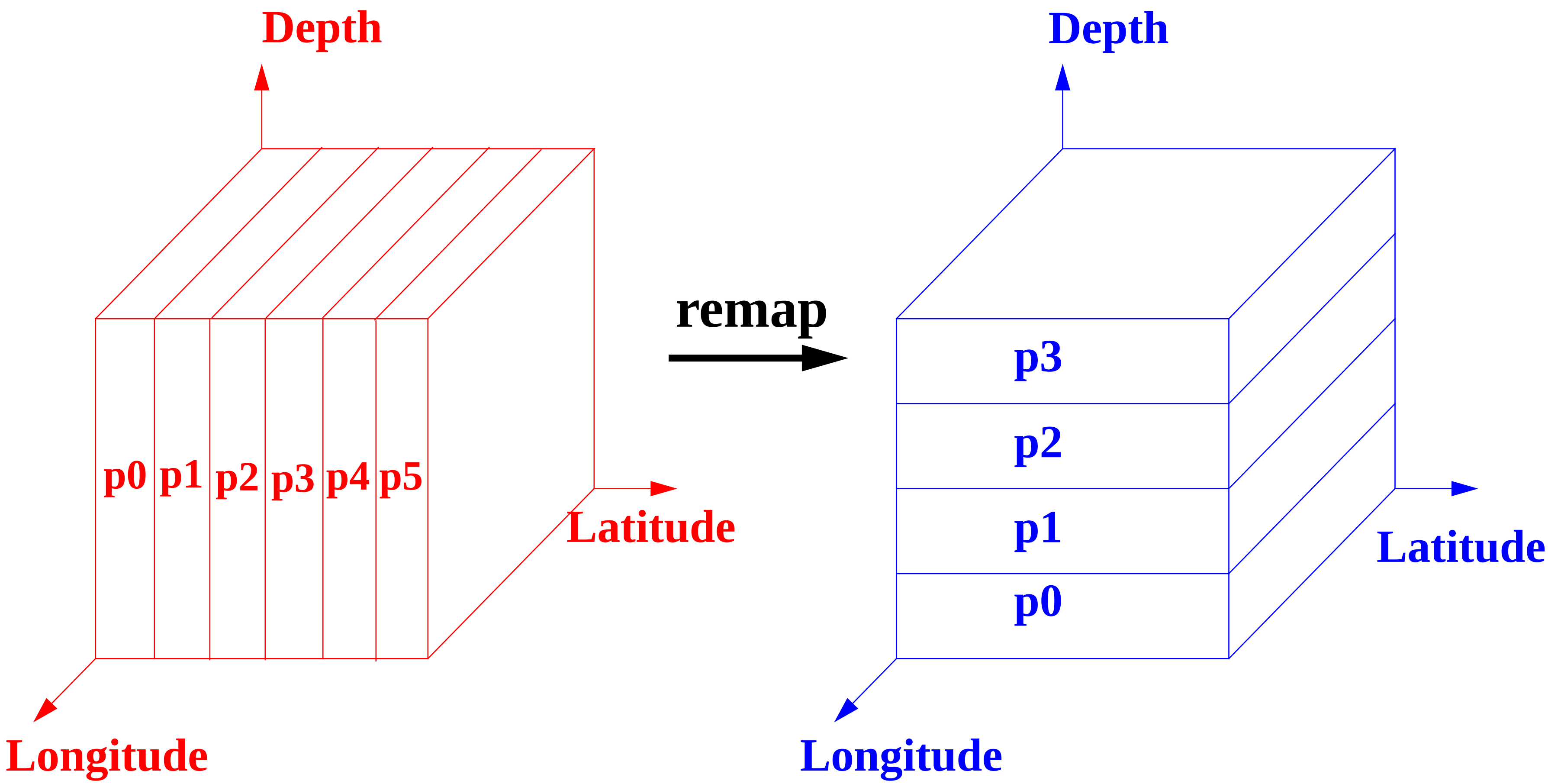


Depth-Split Decomp

Parallel I/O Implementation

- Start with In-Core mode (everything in memory)
- Do 3D arrays, one after another, one at a time.
- Remapping 3D array to depth-split distribution on ioPEs.
  - (# of ioPEs <= # of levels)
  - use in-place remapping algorithm
- Each ioPE write/read the partial 3D array in one shot.
  - Treated as contiguous block in 1D array.
  - Requires only collective I/O for 1D array.
- 2D arrays are done similarly.

Remapping a 3D array from 6 processors to 4 designated I/O processors



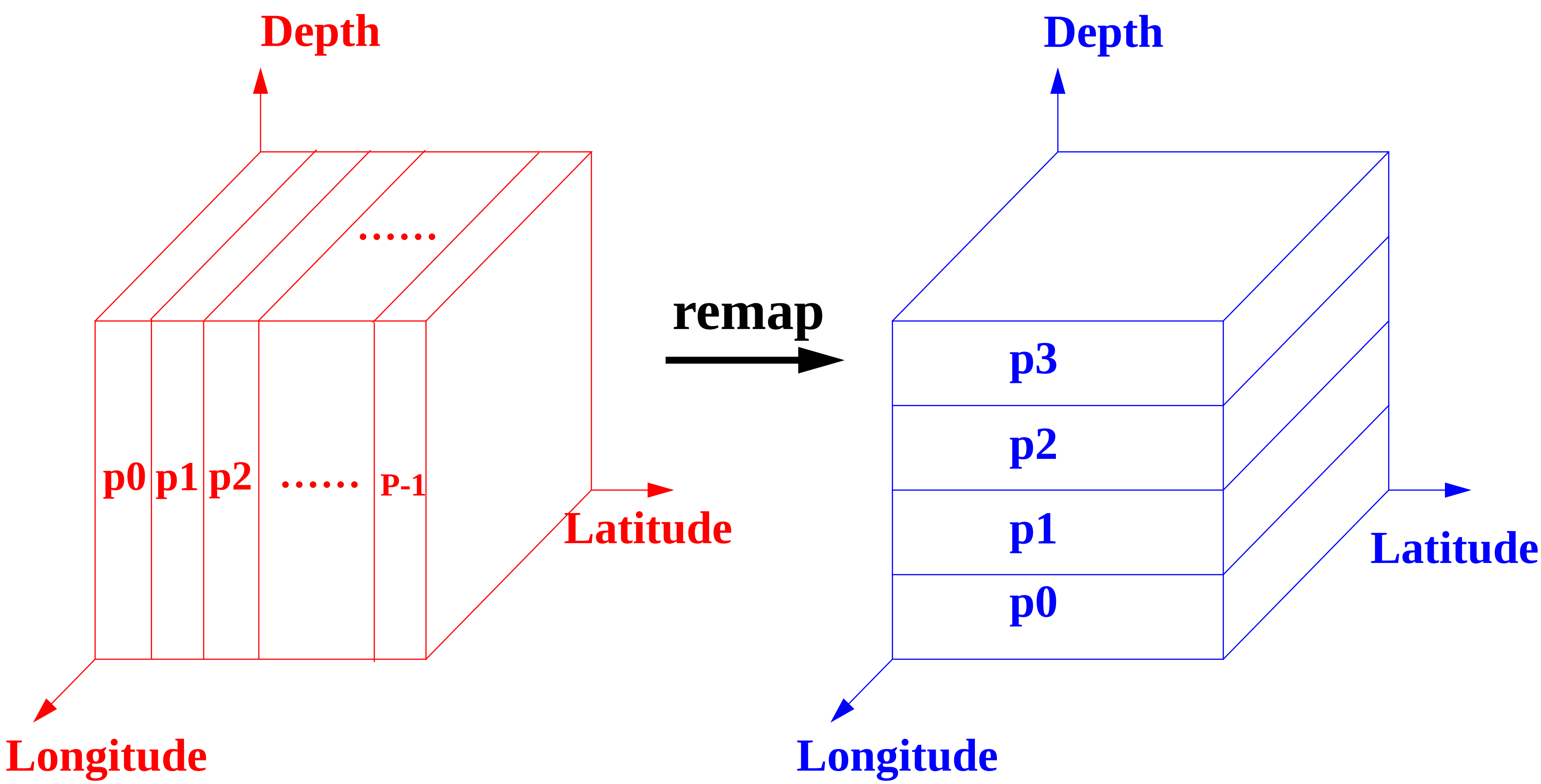
Latitude-Split

A(i, k, j)

Depth-Split

A(i, j, k)

Remapping a 3D array from P processors to 4 designated I/O processors



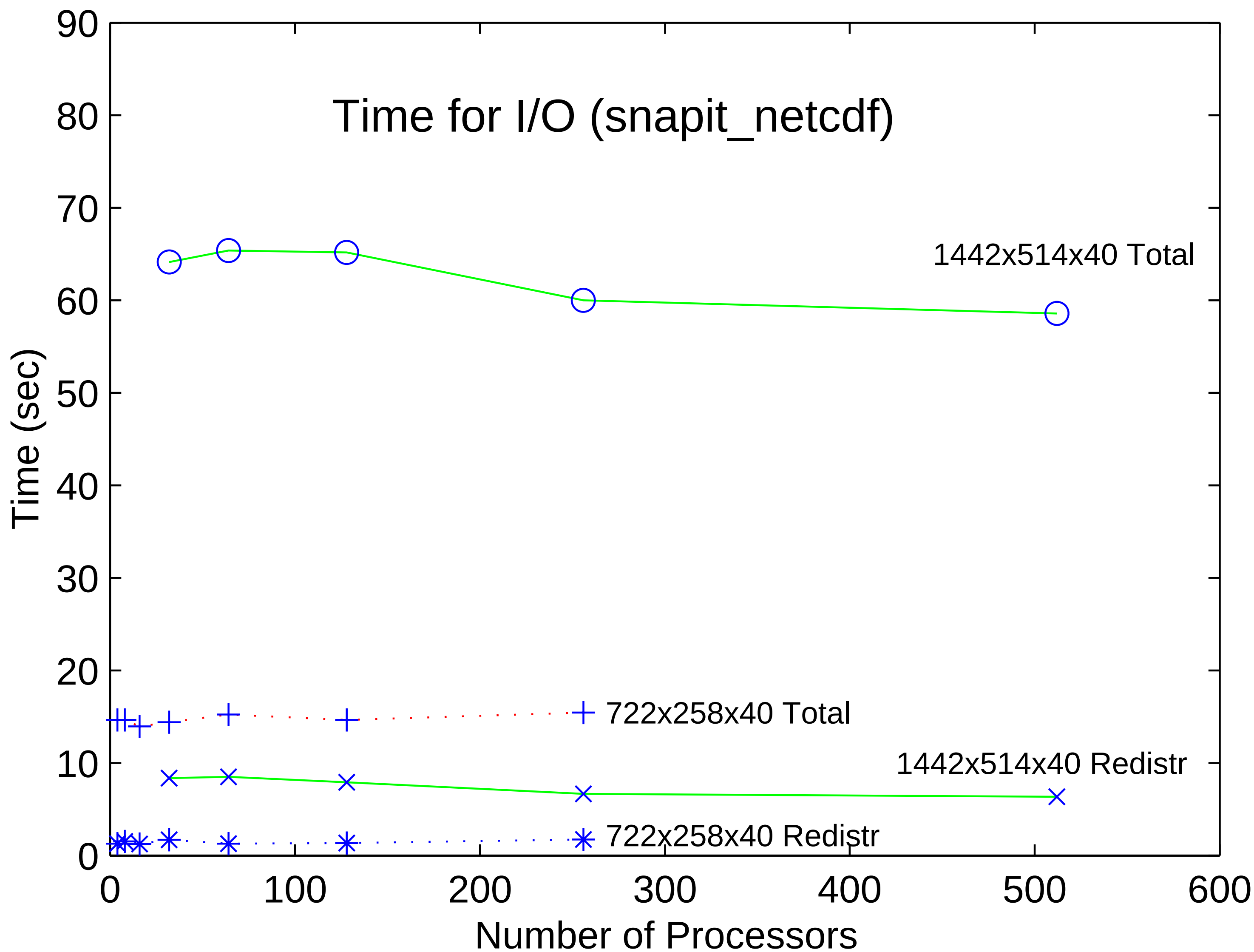
Latitude-Split

A(i, k, j)

Depth-Split

A(i, j, k)

Timing of the Parallel Snapshot I/O for 0.5^{o} and 0.25^{o} Resolutions on T3E with netCDF.



Optimization : Diagnose

- Diag() scales very poorly to large # of processors
- Reason: getunit() is called each time step. It's very time consuming.
- Modified diag() such that it calls getunit() once a simulated day, or a pre-specified interval. This speeds up diag() by a factor of 32!
- Similar modifications are made for relunit().

Summary and Conclusions

- Analysis of data organization and I/O in MOM3
- Out-of-core memory usage mode not suitable
- In-core memory usage mode speedup computing and facilitate I/O
- Sequential netCDF I/O is speedup by 50-fold
- Parallel I/O design and implementation scales well
- An inplace 3D array remapping algorithm developed
- 2D barotropic explicit free surface not scales well.