Parallel Computations in Problems of Climate Modeling*

V. Gloukhov[a]

[a]Research Computing Center, Moscow State University,
NIVC MGU, Vorobjovi Gory, 119992 Moscow, Russia

The paper is focused on different approaches to parallel implementation of climate models on cluster multiprocessors. We present results of applying either MPI or OpenMP to the model components: atmospheric and oceanic blocks. A library of communication routines was developed for the distributed memory approach. It carries out the basic communication operations, such as boundary exchanges and transpositions of decomposed data. The library has quite general interface that makes it useful for parallelization of a wide range of scientific applications calculated on structured grids. In particular, we have examined it on INM AGCM as well as on a set of atmospheric benchmarks which integrate the equations of hydrothermodynamics of atmosphere under the hydrostatic assumption. An OpenMP implementation of the INM Ocean model has been undertaken and tested on few shared memory systems.

INTRODUCTION

The generally accepted definition of climate is the average course or condition of weather at a place usually over period of years as exhibited by temperature, wind velocity, precipitation, etc. Or more rigorously, it is a set of subsequent states of atmosphere, ocean and criosphere, quantified by state variables. Since no direct physical experiment on climate system is possible, scientists use state-of-the-art numerical models to access the role of anthropogenic factors in its formation and changes. Long periods of integration, decades or even centuries, make the problem complex to be processed by a uniprocessor PC or workstation, necessitating the usage of parallel computers, in particular those of cluster architecture, comprised of SMP nodes interconnected with fast network. The concept of parallel processing is quite suitable for climate models because they can be split up into components, for example, into the atmospheric and ocean parts, which interact each other from time to time, through exchange of energy and substance, and are independent in-between. Furthermore, the components, also can be calculated concurrently, providing a substantial degree of parallelism exploitable through shared or distributed memory multitasking.

This paper aims to show few of our experiences in trying different parallelization strategies for an Atmospheric and an Ocean model developed in the Institute of Numerical Mathematics of the Russian Academy of Sciences. We highlight some technical details of these approaches that could be interesting to application developers, not only from the

*Supported by the INTAS projects 00-189, 01-2132 and RFBR projects 01-05-64150a, 03-05-64358a.

area of climate studies but from other fields also. While an application of distributed memory paradigm to the full INM atmospheric general circulation model was described in [1], here we stress only on distributed memory implementation of its dynamical core responsible for integration of the governing equations. And the OpenMP paradigm was tested on the INM Ocean model that constitutes the ocean component of a coupled model [10].

We kept the following structure of the presentation. In section 1, a library of communication routines (communication kernel) is described that performs the basic communication operations on distributed memory: boundary exchanges and transpositions. Written in C the library is intermediate between MPI and application. The communicated data may have quite general format which makes it suitable for parallelization of a wide range of both C and Fortran scientific applications formulated on structured grids.

In particular, the library are invoked by the atmospheric dynamical core to maintain 2D data partitioning over the processors. Extended by the idealized Held-Suarez forcing [2], the dynamics can be used for benchmarking purposes alone. It allows to switch between the explicit and semi-implicit time stepping scheme, as well as between direct and iterative solvers for the Helmholtz equation arising in the semi-implicit scheme, giving an opportunity for direct intercomparison of the methods at different grid resolutions. Section 2 contains an example of such benchmarking carried out on MBC1000M computing system installed at the Joint Supercomputing Center, in Moscow.

Though MPI is quite common programming environment on clusters, other techniques exist facilitating parallelization. In section 3, we present the results of an OpenMP parallelization of the INM Ocean model, which requires less programming efforts than the distributed memory approach, but scalable up to a single SMP node on most systems.

1. COMMUNICATION KERNEL

If a climate model runs on a distributed memory machine and domain decomposition technique is used within its components, each processor computes only particular subdomain of either the atmosphere or ocean. Interactions between the model components and dependencies at the boundaries of subdomains urge the processors to interchange messages. These communications have to be realized as a set of routines.

We are about to characterize a library of such routines, called communication kernel, which is based on MPI. This library was designed for the communications performed within the model components for it has no interpolation features. Moreover, the communicated data are assumed to be stored in multidimensional arrays which dimensions correspond to different spatial dimensions or prognostic variables. Therefore, the library is intended only for the structured grids and rectangular domains which should be decomposed along one ore more spatial dimensions. It performs two major kinds of communications of distributed data: boundary exchanges and transpositions.

Boundary exchange is a local communication, in the sense that every processor exchanges values adjacent to the boundaries of his subdomain only with the neighbors, while the transposition is global, or an all-to-all communication, such that each processor within a group interacts with each. For example, boundary exchanges are inevitable, when derivatives are evaluated by the method of finite differences. But if an operation, for

instance integral summation, requires all the data along a decomposed dimension and is independent along another, which is not partitioned, the data can be redistributed along the second dimension, then the operation is evaluated locally within each processor, and in the end the original distribution is restored. That is exactly what we call transposition.

The boundary exchange routine has the following interface

```
int P_BExchange( void *arr, int nd_array, int *stride,
          int *blklen, int nd_procgrid, int n_edges,
          int *cart2arr, int overlap[][2],
          MPI_Datatype datatype, MPI_Comm comm )

CALL P_BEXCHANGE (ARR, ND_ARRAY, STRIDE, BLKLEN, ND_PROCGRID,
 N_EDGES, CART2ARR, OVERLAP, DATATYPE, COMM, IERR)
INTEGER ND_ARRAY, ND_PROCGRIG, N_EDGES, DATATYPE, COMM, IERR
INTEGER STRIDE(ND_ARRAY), BLKLEN(ND_ARRAY), CART2ARR(ND_PROCGRID)
INTEGER OVERLAP(2, ND_PROCGRID)
```

where `COMM` MPI communicator defining a Cartesian topology, `ARR` pointer to the first element of local subdomain, `ND_ARRAY` number of dimensions of array `ARR`, `ND_PROCGRID` number of dimensions of the Cartesian topology, `CART2ARR` array mapping dimensions of the Cartesian topology into dimensions of the array `ARR`, `STRIDE` dimensions of array `ARR`, `BLKLEN` dimensions of the subdomain, `OVERLAP`, `N_EDGES` width and adjacency degree of boundaries to be communicated, respectively, `DATATYPE` MPI data type of array elements, `IERR` error code.

Figure 1 depicts a local to some processor portion of data, decomposed along directions dir_1 and dir_2, with its halo regions (boundaries). We assume, by definition, that the boundaries B_1^0, B_1^1, B_2^0 and B_2^1 has the first degree of adjacency, while $B_{1,2}^{0,0}$, $B_{1,2}^{0,1}$, $B_{1,2}^{1,0}$ and $B_{1,2}^{1,1}$ the second (the last four will be exchanged in diagonal directions).

The transposition routine has the following interface

```
int P_Transpose ( ndims, void *arr_source, int dim_source,
        int *lblks_source, void *arr_dest,
        int dim_dest, int *lblks_dest, int *stride,
        int *blklen, int *overlap, MPI_Datatype datatype,
        MPI_Comm comm, int period )

CALL P_TRANSPOSE (NDIMS, ARR_SOURCE, DIM_SOURCE, LBLKS_SOURCE,
 ARR_DEST, DIM_DEST, LBLKS_DEST, STRIDE, BLKLEN, OVERLAP,
 DATATYPE, COMM, PERIOD, IERR)
INTEGER NDIMS, DIM_SOURCE, DIM_DEST, DATATYPE, COMM, PERIOD
INTEGER IERR, LBLKS_SOURCE(*), LBLKS_DEST(*)
INTEGER STRIDE(NDIMS), BLKLEN(NDIMS)
```

where `ARR_SOURCE`, `ARR_DEST` pointers to local subdomains within the arrays containing data in original and transposed distribution, respectively, `NDIMS` number of dimensions of both arrays `ARR_SOURCE` and `ARR_DEST`, `DIM_SOURCE` and `DIM_DEST` distributed dimensions

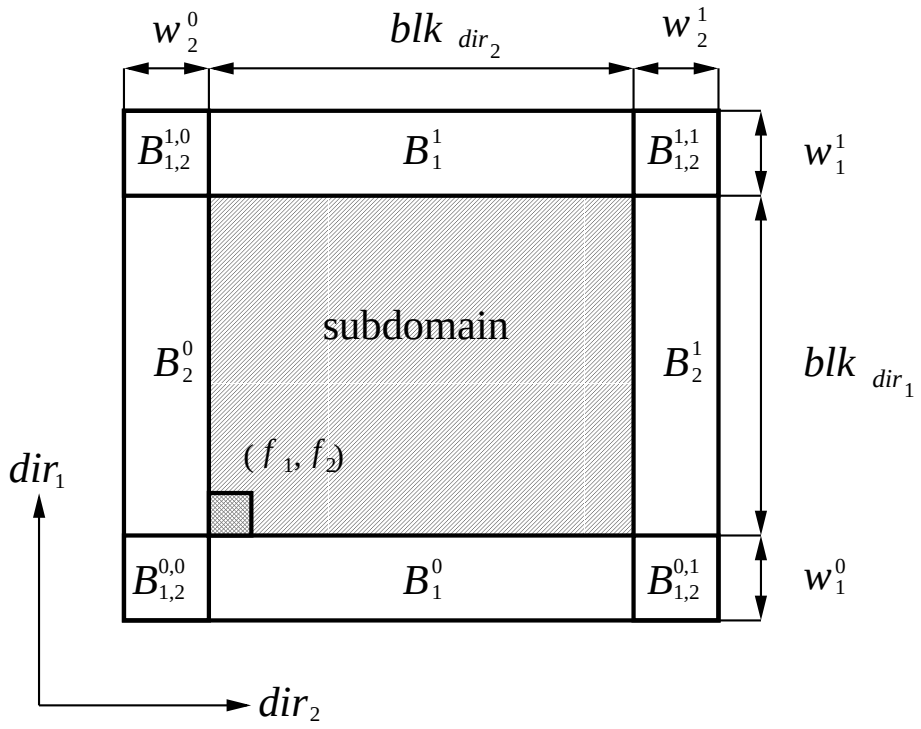

Figure 1. The local subdomain and its halo regions for a 2D data partitioning over the processors.

of the comminicated arrays, respectively, `LBLKS_SOURCE` and `LBLKS_DEST` distribution of dimensions `DIM_SOURCE` and `DIM_DEST`, respectively, `STRIDE` dimensions of the communicated arrays, `BLKLEN` size of local subdomains, `COMM` MPI communicator determining the processors participating in transposition, `DATATYPE` MPI data type of the communicated arrays, `IERR` error code. Parameters `OVERLAP` and `PERIOD` define the interprocessor overlapping of redistributed data and periodicity of dimension `DIM_DEST`. There usage is equiualent to a transposition without overlapping followed by a boundary exchange in the direction of dimension `DIM_DEST`.

Figure 2 represents data in array `ARR_SOURCE` decomposed in direction dir^s (original distribution), while figure 3 shows the same data in array `ARR_DEST` redistributed in direction dir^d (transposed layout).

To transfer data asynchronously the kernel routines call MPI non blocking send-receive functions `MPI_Isend` and `MPI_Irecv` [4], while to form the blocks of data to be transfered they invoke the `MPI_Type_vector` function.

It is worth to mention, that our kernel functionally is similar to the Nearest Neighbor Tool (NNT) developed by Rodriguez, Hart, and Henderson [3].

2. EULERIAN DYNAMICAL CORE

Robustness of the communication kernel described above has been tested on the dynamical core, used in INM AGCM [5], which integrates a system of primitive equations

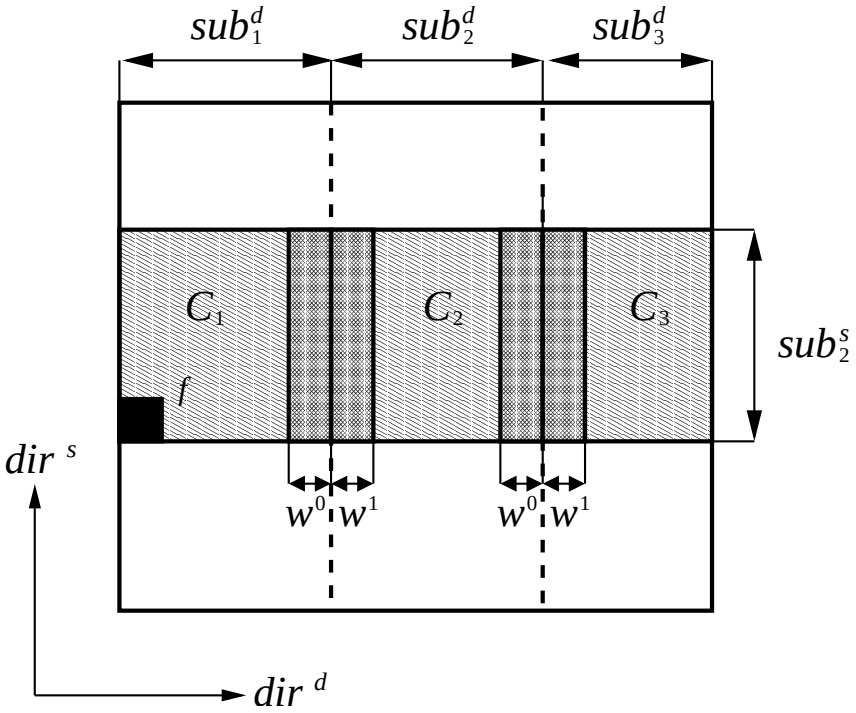

Figure 2. Data distribution before transposition.

of hydrothermodynamics of the atmosphere in the Boussinesq approximation under the hydrostatic assumption [6]. Written in vertical σ-coordinate, the system is discretized on a staggered Arakawa C-grid [7]. Our realization allows user to choose between explicit or semi-implicit time integration scheme. Since most calculations performed by the AGCM are very dependent in the vertical direction, we applied checkerboard partitioning, distributing data along both longitude and latitude, over two-dimensional Cartesian topology of processors. Evaluation of the explicit tendencies at a grid point requires the values of prognostic variables from some neighborhood of that point, necessitating a boundary exchange. Also a transposition is required to perform Fourier filtering of the fields near the poles.

The semi-implicit scheme increases time steps 4–5 times, but leads to a discrete two-dimensional Helmholtz-like equation that have to be solved on every vertical level each time step. The matrix of the arising linear system is structured and can be represented as $I - q_k[D_\varphi^2 \otimes M_\lambda + (D_\varphi M_\varphi) \otimes I_\lambda]$, where q_k are height dependent coefficients, I_λ is a unit matrix, D_φ is diagonal, M_φ tridiagonal, and M_λ cyclic tridiagonal.

There are few fast direct methods of solving the system with structured matrix, like we have, which take advantage of the matrix structure. We exploit a direct method involving the fast Fourier transform (FFT) along longitude and tridiagonal Gaussian eliminations along latitude. However, both longitudes and latitudes are distributed, and the alternative here is whether to use distributed algorithms of FFT and tridiagonal Gaussian elimination or transpose the data twice, first, in longitude-latitude plane and then, after completion of FFTs, in latitude-longitude plane to perform Gaussian sweeps. Our experience convinces

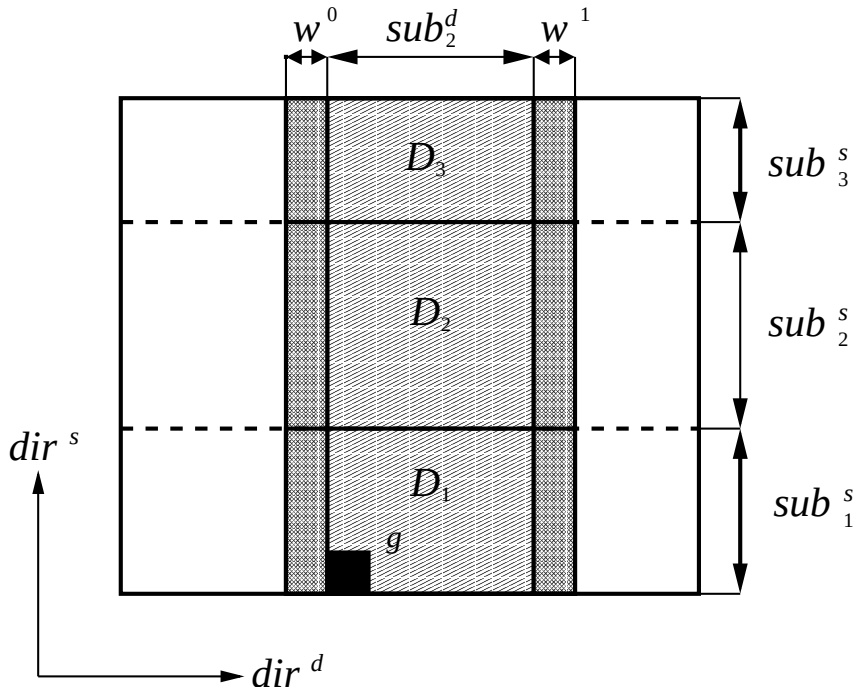

Figure 3. Transposed data.

us that the second approach is more efficient unless the length of transformed vectors is not big. Thereafter, we incorporated double transposition in direct Helmholtz equation solver.

To apply iterative solver we made use of the PIM 2.2 package [8], which we found to be useful. It includes a number of iterative methods for symmetric and nonsymmetric systems. As our matrix is nonsymmetric, we inclined to the Bi-CGSTAB method with the stopping criteria $||r_k|| < \varepsilon||b||$, where r_k is the residual, b is the right hand side, and $\varepsilon = 10^{-2}$. The only thing supplied by us to PIM was a matrix-vector multiplication subroutine, that invokes the boundary exchange communication routine.

Table 1 contains an example of benchmarking performed with the Eulerian dynamical core on MBC1000M computing system. One can see, that the semi-implicit scheme with the direct solver more scalable than that with the iterative solver and faster than the explicite scheme, at given resolution.

3. OPENMP IMPLEMENTATION OF THE INM OCEAN MODEL

The INM Ocean model [9] is based essentially on the same governing equations as the atmospheric model, with the difference that the salinity is incorporated as a prognostic variable. As a component of the coupled model [10] it performs about 10% of overall computations. The rigid lid condition at the surface allows to introduce the barotropic stream function. The numerical implementation makes use of splitting into physical processes, as well as along spatial coordinates, facilitating application of the efficient implicit

Table 1
CPU time (above) and speed-up (below) of one day forecast (in sec) of the Eulerian dynamical core at resolution $256 \times 128 \times 16$ on a given number of processors of MBC1000M computing system. EXPL explicit scheme, GAUSPH semi-implicit scheme with the direct solver, ITER semi-implicit scheme with the iterative solver.

# of procs	1	2	4	8	16	32	64	128	256
EXPL	602.	367.	190.	103.	54.	43.	24.	20.	18.
	1.0	1.6	3.2	5.8	11.0	13.9	25.4	30.6	33.5
GAUSPH	233.	156.	100.	69.	53.	34.	17.	11.	10.
	1.0	1.5	2.3	3.4	4.4	6.9	13.8	20.7	23.4
ITER	306.	178.	99.	53.	35.	35.	31.	31.	32.
	1.0	1.7	3.1	5.8	8.5	8.6	9.7	9.9	9.5

algorithms. Introduction of the relative depth coordinate makes the computational domain cylindrical. Though the horizontal grid is uniform in spherical coordinates and fields are stored in multidimensional arrays, which dimensions corresponding to spatial coordinates, only wet points are computed and the processed domain has rather complex form in horizontal plane.

The shared memory approach is attractive, mainly, because it does not involve such profound restructuring of code, as MPI does. Secondly, it is suitable for the Ocean model, because the dynamic scheduling, if enabled, yields good load balancing even for an ugly domain.

The model parallelization was accomplished by supplying the `PARALLEL DO` directive to major loops. Red-black ordering of unknowns was introduced in the SOR method that calculates the barotropic stream function. Ultimately, speedup 2.6 was reached on a 4-processor Itanium II system (Table 2). A distributed memory version of the model, obtained by A.S. Rusakov [11] outperforms the OpenMP version. On 8 CPUs of MBC1000M it revealed speed-up of 4.9 which, however, was estimated without writing some output files.

Table 2
CPU time of the OpenMP parallel version of the INM Ocean model one year runs.

# of CPUs	1	2	4	8
IBM Regatta, p670	16m47s	11m28s	8m39s	7m16s
Intel Itanium II	15m46s	9m46s	5m58s	
Sun UltraSparc II	1h38m24s	1h5m48s	48m48s	

ACKNOWLEDGMENTS

We would like to thank the Joint Supercomputer Center for providing there computing facilities which include the first Russian Teraflop machine MBC1000M. Also support

of Russian-Indian Centre for Advanced Computing Research is appreciated. This work would not be possible without assistance of E.M. Volodin, N.A. Diansky, and M.A. Tolstykh, all from INM RAS.

REFERENCES

1. V. Gloukhov, Parallel implementation of the INM atmospheric general circulation model on distributed memory multiprocessors, Lecture Notes in Computer Science 2329 (2002) 753–762.
2. I.M. Held and M.J. Suarez, A proposal for the intercomparison of the dynamical cores of atmospheric general circulation models, Bull. Am. Meteorol. Soc. 73 (1994) 1825–1830.
3. B. Rodriguez, L. Hart, and T. Henderson, Parallelizing operational weather forecast models for portable and fast execution, Journal of Parallel and Distributed Computing 37 (1966) 159–177.
4. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, MPI: The Complete Reference, The MIT Press, Cambridge, MA, 1998
5. V.A. Alexeev, E.M. Volodin, V.Ya. Galin, V.P. Dymnikov, V.N. Lykossov, Simulation of the present-day climate using the INM RAS atmospheric model. The description of the model A5421 (1997 version) and the results of experiments on AMIP II program. Institute of Numerical Mathematics RAS, Moscow (1998) (reg. VINITI 03.07.98 No. 2086-B98)
6. G.I. Marchuk, V.P. Dymnikov, V.B. Zalesny, V.N. Lykossov, V.Ya. Galin, Mathematical Modeling of the Atmosphere and Ocean, Gidrometeoizdat , Leningrad, 1984 (in Russian)
7. A. Arakawa, V.R. Lamb, Computational design of the basic dynamical processes of the UCLA general circulation model, Methods Comput. Phys. 17 (1977) 173–265.
8. R.D. Cunha and T. Hopkins, PIM 2.2 The parallel iterative methods packages for systems of linear equations. User's guide. Internal Report 2–94 of the Computing Laboratory, UKC.
9. N.A. Diansky, A.V. Bagno, and V.B. Zalesny, Sigma model of global ocean circulation and its sensitivity to variations in wind stress, Izvestiya, Atmospheric and Oceanic Physics, 38 No. 4 (2002).
10. N.A. Diansky, E.M. Volodin, Reproducing the present day climate using a coupled atmosphere-ocean general circulation model. Izvestia, Atmospheric and oceanic physics 38 No. 6 (2002).
11. A.S. Rusakov, N.A. Diansky, Parallel ocean general circulation model for distributed memory computer systems, Parallel CFD 2003, Book of abstracts.