A METHODOLOGY FOR DEVELOPING HIGH PERFORMANCE COMPUTING MODELS: STORM-SCALE WEATHER PREDICTION

Nikos Chrisochoides,[*] Kelvin Droegemeier,[**] Geoffrey Fox,[*] Kim Mills[*], Ming Xue[**]

[*]Northeast Parallel Architectures Center, Syracuse University

111 College Place, Syracuse, NY, 13244-4100

[**]Center for Analysis and Prediction of Storms, University of Oklahoma

100 East Boyd, Norman, OK 73019-0628

corresponding author: Kim Mills (kim@npac.syr.edu)

ABSTRACT

A methodology for developing future generations of a storm-scale weather prediction model for Massively Parallel Processing is described. The forecast model is the Advanced Regional Prediction System (ARPS), a three-dimensional, fully compressible, non-hydrostatic predictive model. In the short term, the computational goals include developing a portable, scalable model for distributed memory SIMD and MIMD architectures, while preserving a high degree of modularity to support rapid design and validation, maintainability, educational goals and operational testing. Longer term computational goals include a parallel adaptive mesh refinement scheme. A FortranD/High Performance Fortran version of the ARPS provides portability in the current version of the model, and supports future model research goals.

INTRODUCTION

We present a survey of issues by the Center for Analysis and Prediction of Storms (CAPS) at the University of Oklahoma, and the Northeast Parallel Architectures Center (NPAC) at Syracuse University for applying parallel computing technology to operational storm-scale numerical weather prediction.

CAPS is a National Science Foundation Science and Technology Center with the mission of demonstrating the practicability of storm-scale numerical weather prediction. A centerpiece of this effort is the development of a multi-scale, scalable/parallel model–the Advanced Regional Prediction System (ARPS)– that is based on advanced numerical techniques and treatments of physical processes, and that can be applied to weather phenomena on scales ranging from a few kilometers and tens of minutes (individual thunderstorms) to hundreds of kilometers and several hours (storm complexes and mesoscale systems) (Droegemeier 1990; Lilly 1990). Due to the sensitivity of small-scale weather to local forcing, and the future availability of a national network of large bandwidth WSR-88D (formerly called NEXRAD) Doppler radars, from which the ARPS will obtain its initial conditions, CAPS views its research as a prototype foundation for self-contained regional prediction centers. A numerical weather prediction model running on massively parallel computers is central to this plan.

NPAC is a parallel computing research center supported by industry, government, and university funding with the mission of applying parallel computing technology to large scale problems in science and industry. The FortranD portable, parallel compiler is a key technology development project being carried out at NPAC and Rice within the Center for Research in Parallel Computation. FortranD is a research prototype of industry standard High Performance Fortran (High Performance Fortran Forum 1992), and is central to our plans for developing a high performance ARPS.

Our principal goal for the CAPS/NPAC collaboration is to attain problem sizes and model execution rates with the ARPS that allow us to investigate new scientific problems and, in the future, to support operational prediction of storm-scale events. The model must be: portable across single instruction multiple data (SIMD) and multiple instruction multiple data (MIMD) distributed memory architectures, workstation-based distributed computing networks, and shared-memory vector machines; scalable over system sizes ranging from tens to thousands of nodes; and easily interfaced to related modules for data assimilation, visualization, and post-processing. Two hour operational forecasts will require the collection of approximately 30 minutes of observational data (100 MBytes/radar x 10 or so radars), the running of a computationally intensive data assimilation model, and finally, the forecast model. A two hour forecast now takes approximately one hour to produce for a domain that is sixteen times smaller than that desired for operational use, resulting in a loss of one hour of effective forecast time and even more time when one considers the time required for data ingest, quality control, assimilation, and forecast product dissemination to the public. Our current efforts are directed toward shortening the time to run the forecast system.

Our methodology for developing a high performance implementation of ARPS is based on High Performance Fortran (HPF). We are currently producing a set of ARPS implementations in various Fortran dialects (Fortran77, Fortran77+message passing, Fortran90, and HPF) in order to evaluate parallel software issues. A HPF version of ARPS will support the immediate goals of model portability and scalability, as well as the linking of future ARPS development efforts with the development of HPF. Through HPF, we expect to apply the latest results in parallel computing research to ARPS, including numerical methods, parallel algorithms, and grid adaption techniques.

THE CAPS ADVANCED REGIONAL PREDICTION SYSTEM (ARPS)

The ARPS forecast model is a grid-based, three-dimensional, fully compressible, nonhydrostatic predictive model. The primitive equations of fluid flow are solved using the finite difference approach with explicit time differencing on a spatially-staggered grid mesh (CAPS 1992). CAPS believes model domains on the order of 1000 x 1000 x 20 km will be required for regional-scale prediction (1 km horizontal by 0.5 km vertical). When related tasks are ignored, such as data observation, data assimilation, and post-processing, a forecast model of this size is estimated to require approximately 10 GigaFlops performance to model the weather in real time. Producing a two hour forecast in 1.2 minutes (model 100 times faster than the weather) would require 1 TeraFlops of performance.

The ARPS model is designed to exploit the parallelism and modularity inherent in the governing hydrodynamical equations in order to yield a code that is naturally scalable and adaptable to wide varieties of parallel computer architectures (CAPS, 1992). For example, a set of differencing and averaging operators is used as a building block throughout the code to calculate mixing, advection, and Coriolis terms in the momentum equations. Our short term goal is to develop a portable, scalable ARPS model for distributed memory SIMD and MIMD architectures, while preserving a high degree of modularity to support rapid design and validation, maintenance, educational and operational testing goals.

Future plans for model development include: improved numerical methods based on higher order differencing stencils; semi-Lagrangian integration schemes to lengthen the model timestep without degrading the accuracy of the solution (Staniforth and Cote 1991); use of a piecewise parabolic method as an alternative to conventional gridpoint methods to model flows in regions with sharp gradients (Carpenter et al. 1990); and grid adaption techniques to provide locally high spatial resolution in appropriate regions.

Semi-Lagrangian techniques (e.g., Staniforth and Cote 1991) introduce more complex communication requirements because the future state of a given grid point is no longer a function of a neighboring grid point at the previous time step, but rather is a function of an entire set of grid points in the vicinity of the given grid point. Pure Lagrangian methods have been previously developed for parallel systems (Trease and Cerutti 1992), and we expect to take advantage of this earlier work. Piecewise parabolic methods have been implemented for meteorological flows in one dimension (Carpenter et al. 1990); expanding this method to three dimensions, and running on parallel systems is the subject of ongoing research (Droegemeier et al. 1992).

The challenge to our research team is to produce a model for operational forecast purposes that outperforms a uniform fine-mesh model in both the quality of the solution as well as the ratio of wallclock time to model simulated time. We base our approach on a HPF version of ARPS that supports current objectives (portability) as well as future goals (improved numerical methods), but most notably, the development of parallel mesh refinement techniques. This approach complements related collaborative efforts between CAPS, the Army High Performance Computing Research Center at the University of Minnesota, and the Supercomputer Computations Research Institute at Florida State University.

FORTRAND/HIGH PERFORMANCE FORTRAN

FortranD is a Center for Research on Parallel Computation (CRPC) software development project being carried out at Rice and Syracuse Universities with NSF and DARPA funding. The goal is to produce a machine-independent parallel programming model. FortranD is built upon Fortran77 and Fortran90, with extensions for expressing parallelism in the problem application and mapping the problem to a specific machine architecture. "Decomposition" and "alignment" directives define the size and dimensionality of the problem arrays, as well as alignments between arrays. "Distribute" directives specify the mapping of a problem to a physical machine, taking into account characteristics such as topology, the communication network, the size of local memory, and the number of processors. The user is involved in this process by design since he or she best understands the application code; FortranD attempts to take attempts to exploit this information. High Performance Fortran (HPF) was established as the industry standard data parallel Fortran in December, 1992, with a subset of Fortran90D built into HPF.

The development of FortranD is based on a synergism between real world problems and software development. We view parallel software as the most important obstacle to developing parallel applications, and use evaluations of challenging application problems to define extensions and improvements required in FortranD. FortranD, and current data parallel languages, provide support for expressing regular and loosely synchronous problems. The language support needed to implement our proposed parallel adaptive mesh techniques is also available. By developing the ARPS model for both MIMD and SIMD machine architectures, and data parallel and message passing software architectures, we will be able to identify strategies for ARPS model development, and the extensions needed in FortranD to support these strategies. Thus, we have incorporated ARPS into our FortranD benchmark suite of industrial and scientific application codes.

COMPUTATIONAL MODELS

We will use data parallel and explicit message passing computational models to implement ARPS on massively parallel processor systems. Data parallel code tends to be easier to understand, write, and debug than message passing codes. Although message passing works for a large set of applications on MIMD machines, it is not completely portable. The programmer must explicitly express message passing, overlap of communication and calculation, and decomposition choices. Thus, although message passing works, a more easily-implemented approach is desirable. Fortran77 + message passing provides portability to all MIMD machines, and Fortran90D/HPF provides portability to all SIMD and MIMD machines.

Message passing model

The ARPS model is grid-based and uses a second-order finite differencing scheme. Issues relevant to an explicit Fortran 77 plus message passing model include domain partitioning and update ordering, interprocessor communication, and scalability–the ratio of calculation to communication time, and resulting efficiency of the algorithm as a function of problem size and number of processors. The uniform structure of the current version of ARPS leads to a simple decomposition strategy (Figure 1). For example, using our 32 node Connection Machine-5 and an ARPS model of size 32x32x32 grid cells, we will define a 4x4x2 processor grid. A model subdomain of 8x8x16 grid cells will be assigned to each processor.

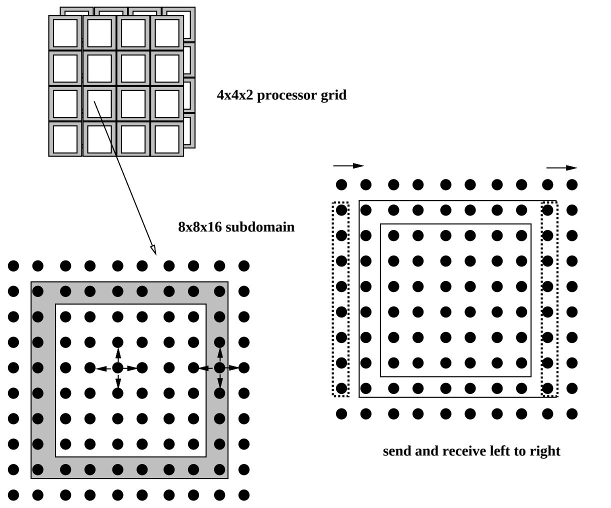


Figure 1: Decomposition of finite difference problem for multiple processors, (see Fox et al. 1988. Solving Problems on Concurrent Processors. Prentice Hall, Englewood Cliffs, NJ. p. 121.)

The discrete difference and averaging operators currently used require only nearest neighbor communication. Grid points within a subdomain require only data stored in local processor memory, while grid points within the subdomain boundary must access data stored in a neighboring processor's memory, thereby requiring interprocessor communication. All grid points on a face, or plane, of a three-dimensional subdomain will communicate with the neighboring processor, and all processors will send and receive messages concurrently.

We are following the model development approach of CAPS by emphasizing the use of fine grain communication modules that are easy to understand and modify (CAPS 1992). While implementation of the ARPS as an explicit message passing model is not yet complete, preliminary results indicate that communication time accounts for a small percentage of model run time on the CM-5, and we expect very good performance for the message passing version of the code.

Data parallel model

We will divide our analysis of the ARPS code into dynamic modules and physics modules. The dynamics modules are composed of differencing and averaging operators, and are used, for example, to calculate turbulent mixing terms for momentum equations, or advection of potential temperature. Only nearest neighbor communication is presently required. One of the physics modules we examined applies Kessler warm rain microphysics to water vapor, liquid water, and temperature fields. These calculations are applied to vertical columns within the model domain, and are computationally independent. The regular, synchronous structure of the ARPS model makes it well suited as a data parallel model and a Fortran90 version of ARPS is nearly complete. We observed speedups of 300 times for differencing operators, and 70 times for a physics module when comparing model run times for an 8K Connection Machine-2 and a SUN4.

PARALLEL MESH DEVELOPMENT

A major challenge in Numerical Weather Prediction (NWP) is the accurate representation of localized, small-scale features (e.g., clouds) embedded in larger-scale flows. Techniques to accomplish this must be computationally efficient in a high-performance forecast model, and thus the use of a uniformly high resolution throughout the model domain is an ineffective solution for present-day machines. The most common solution involves adapting the grid in such a way that increased spatial resolution is provided in regions of large solution error (e.g., large gradients or regions where the gradient changes rapidly). A number of adaptive schemes exist, and in this paper we consider solvers based on structured meshes and refinement techniques based on a combination of nested grids and equidistribution approaches. The nested grids are generated using domain decomposition methods (tile approach), with their points redistributed using an attraction-repulsion (AR) method which we parallelize using systolic algorithms.

Nested Grids

The basic idea of the *nested grid* approach for adaptive grid refinement (Berger and Oliger 1984) involves creating and deleting finer subgrids in a background mesh in order to obtain a given level of accuracy with a minimal number of grid points. Figure 2 depicts a nested grid in a shock region. The nested grid approach combines the advantages of both global and pointwise refinement approaches. Similar to pointwise refinement approaches, nested grids increase the resolution of a coarse grid only within a neighborhood of marked grid points which define regions of large solution error (e.g., Skamarock 1989). At the same time, data structures remain similar to those used by global refinement approaches. Nested grids are considered the logical choice for adaptive weather models which require simple and fast data management routines, together with optimum data storage schemes, in order to solve realistic size problems (Skamarock et al. 1989) .

In the following paragraphs, we outline the basic algorithm for creating nested grids and discuss associated implementation difficulties on distributed memory MIMD/SIMD machines. We then describe how

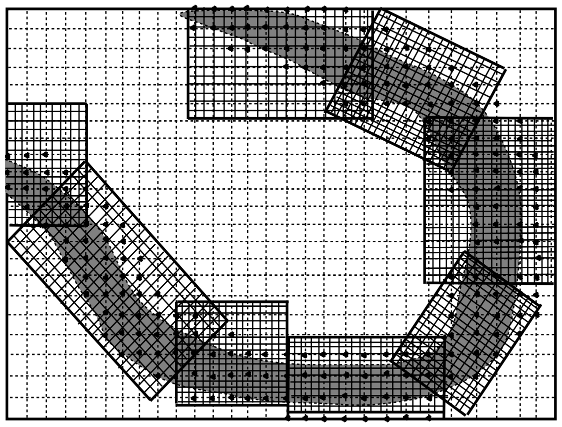


Figure 2: Nested grid in a shock region. The gray area indicates the high gradient regions.

domain decomposition methods resolve some of the problems related to parallel implementation of nested grid algorithms.

The grid generation algorithm requires as input an initial coarse grid and a list of marked grid points on a coarse grid. The algorithm returns as output rotated, rectangular, finer subgrids containing marked coarse grid points while minimizing the total area of the subgrids. The algorithm consists of the following three steps (Berger and Oliger 1984):

1. separate the marked grid points into clusters,
2. fit a rotated rectangular grid to each cluster, and
3. repeat steps 1 and 2 to minimize the area of the subgrids that is unnecessarily refined.

Two major disadvantages of this algorithm are: (i) the limitations of clustering heuristics in the first step of the algorithm, and (ii) the programming complexity required to achieve efficient data parallel algorithms. Some of the clustering heuristics do not work well on all input data (Berger 1985), and other heuristics require complicated data structures in order to compute the minimum spanning tree of the marked, coarse grid points. Moreover, the efficient implementation of the clustering heuristics on distributed memory MIMD/SIMD machines is even a more difficult problem. Referring to the second disadvantage, the various types of interactions between adjacent or overlapping subgrids increase the code complexity and the communication overhead required to ensure the consistency of the distributed data structures. Arbitrarily oriented nested subgrids make impractical domain decomposition methods in developing data parallel models. The domain decomposition partitions the computation into unbalanced tasks, and the resulting mapping creates unbalanced workloads among processors. For nested grids, it is better to map computations based on a weighted-partitioning of the nested grid levels (McCormick and Quinlan 1989).

The following approach helps us to resolve the clustering and code complexity issues. In this approach (Gropp and Keyes 1990), the generation of nested grids is based on the decomposition of the domain, D, into rectangular subdomains, $D_{i,j}$, ("tiles") so that :

$$D_{i,j} \cap D_{k,l} = \emptyset \quad and \quad \cup_{i,j} D_{i,j} = D,$$

$$\begin{aligned} \partial D \cap \partial D_{i,j} &= \emptyset & if \;\; D_{i,j} \;\; interior \\ &= closed \;\; interval \;\; if & D_{i,j} \;\; boundary, \end{aligned}$$

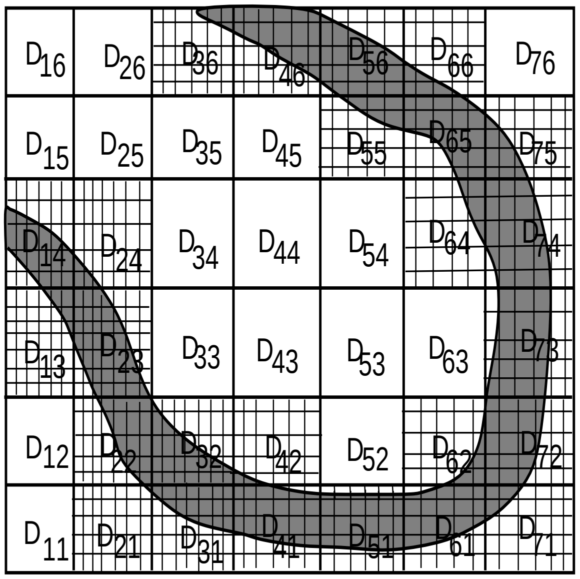
and

$$\begin{aligned} \overline{\partial D_{i,j}} \cap \overline{\partial D_{k,l}} &= \emptyset \qquad if \;\; | i-k | > 1 \; and \;\; | j-l | > 1 \\ &= point \;\; if \;\; | i-k | = 1 \;\; and \;\; | j-l | = 1 \\ &= edge \quad if \;\; | i-k | = 1 \; and \;\; | j-l | = 0 \\ &\qquad or \quad | i-k | = 0 \;\; and \;\; | j-l | = 1 \end{aligned}$$

where ∂D, $\partial D_{i,j}$ are the boundary of the domain D and subdomain $D_{i,j}$ respectively, and $\overline{\partial D_{i,j}}$ is the closure of the boundary of the subdomain $D_{i,j}$ (Rudin 1976) The domain decomposition is based on a coarse grid over the domain D. The coarseness of the grid affects the load balance of the computation since we are not allowed to map the computation associated with a tile onto more than one processor. Figure 3 depicts a nested grid (or quasi-uniform grid) based on this approach. A detailed description on the interaction between adjacent tiles is presented in Gropp and Keyes (1990). Note that the tile-approach eliminates use of clustering heuristics and simplifies the programming complexity and data structures for managing the interfaces between tiles. The only disadvantage of this method is that the quasi-uniform grid does not automatically align with sharp gradient features (e.g. fronts) (Figure 3). We try to eliminate this shortcoming in the next subsection which addresses the grid alignment problem. (Note that non-aligned sub-grids may be desirable for some physical problems.)

Equidistribution techniques

In this subsection, we review equidistribution techniques used to locally align grid points of each tile, and present a systolic algorithm for efficient parallel implementation of the AR technique on distributed memory MIMD machines. Finally, we conclude by presenting an adaptive parallel grid generation method that is a combination of the tile-approach and the parallel AR equidistribution technique.



D16 D26 D36 D46 D56 D66 D76
D15 D25 D35 D45 D55 D65 D75
D14 D24 D34 D44 D54 D64 D74
D13 D23 D33 D43 D53 D63 D73
D12 D22 D32 D42 D52 D62 D72
D11 D21 D31 D41 D51 D61 D71

Figure 3: Nested grids generated by domain decomposition into rectangular subdomains or tiles.

Equidistribution techniques are based on reducing the error by distributing grid points so that some positive weight function, $w(x)$, is "equally" distributed over the computational space (Brackbill and Saltzman 1982; Thompson et al. 1985; Paine 1991; Dietachmayer and Droegemeier 1992; Fiedler and Trapp 1993).

According to Babuska and Reinboldt (1978), the point distribution is asymptotically optimal if the weight function represents an error measure which is stable under perturbations of the point distribution. Russell and Christiansen (1978) reviewed a number of different weight functions, including:

$$w(x) = \Delta x^k \;||\; u^{(k+1)} \;||$$

$$w(x) = \sqrt{1 + || \; u_x^2 \; ||}$$

$$w(x) = \Delta x^k \;\; times \;\; the \;\; residual, \;\; e.t.c.$$

Here $u^{(k)}$ indicates the kth derivative in the x-direction of the solution which, in general, is the solution vector of a linear system of equations, and Δx indicates the local grid spacing.

In two and three dimensional spaces, we must couple the adaptation of the grid in the different directions in order to maintain, along with the optimal grid distribution, some degree of orthogonality and smoothness. For multiple space dimensions, there exist two approaches for implementing equidistribution techniques. The first is based on *control functions* (CF), and the second is based on a minimal principle.

The mathematical formulation of the minimum principle in the continuous form is based on variational (V) methods, and in the discrete form is based on linear minimization problems with linear constraints. The discrete problems are much harder to solve than the unconstrained problems that occur in the continuous case. A comprehensive review of the CF and V techniques, for the continuous case, is presented in Thompson et al. (1985)

In the continuous case, both the CF and V approaches reduce the point distribution problem into a problem of solving a linear or nonlinear system of equations. These systems may be solved iteratively using the successive over-relaxation (SOR) or Jacobi methods, combined with red-black ordering. The linear or nonlinear system comes from the discretization (by finite-difference method, e.g., 5-point star stencil) of a system of elliptic partial differential equations defined on a rectangular computational space. The SOR method combined with red-black ordering can be implemented by using two Jacobi iterations through the unknowns, each on roughly half of the unknowns (Hageman and Young 1981). Note that the parallel implementation of one Jacobi iteration on distributed memory MIMD/SIMD machines requires nearest neighbor communication (i.e., four messages), along with a number of global reduction operations for checking stopping criteria and accelerating convergence (Chrisochoides et al. 1992). For time-dependent models such as those used for weather prediction, it should be noted that the solution and grid evolve together at each time level. Redistribution of grid points is a relatively expensive process, and may require more CPU time than the actual solver (Thompson et al. 1985; Paine and Droegemeier 1991), though more efficient methods have recently been developed (Fiedler and Trapp 1993).

Another technique for dynamically adaptive grid distribution is presented in Anderson and Rai (1982). This technique, similar to the variational methods, is based on the assumption that the best grid is the one that has the same error at each point; instead of moving the grid points through a solution of partial differential equations, dynamic adaptive grid distribution moves grid points under the influence of mutual attraction or repulsion.

In the following paragraphs we describe the basic idea of the attraction-repulsion method in one dimension and our scalable algorithm for the implementation of the AR method on distributed memory MIMD/SIMD machines.

Let $|| w_i ||$ be the magnitude of the error, or of the variation of the solution at the ith grid point, and let $|| w ||_m$ be the arithmetic mean of the magnitude of this measure over all grid points. In order to uniformly distribute the error of the numerical solution (i.e., weather prediction) on a grid with fixed number of mesh points, we must increase the number o points in regions of large error (i.e., $|| w_i || \geq || w ||_m$) and reduce the number of points in regions of small error (i.e., $|| w_i || \leq || w ||_m$). Such a grid can be constructed by forcing the points i, with $|| w_i ||$ larger than $|| w ||_m$ to attract other points, and the points j, with $|| w_j ||$ smaller than $|| w ||_m$, to repel other points. Since the purpose of the AR method is to capture local variations of the error or of the solution, and to force local alignment of the grid, we assume that distant points have little influence on each other. This can be enforced by a $1/r^n$ law as an attenuation factor, where r is the distance between the points i and j.

Every point moves with a velocity whose magnitude and direction depend upon the magnitude of the measure w of all other points. Thus for two points i, j the grid velocity at the ith point is given by

$$V_{i,j} = K \frac{|| w_j || - || w ||_m}{r^n}$$

where n is the power that controls the attenuation, and K is a proportionality constant. For a problem with N points the velocity at the ith point is given by

$$V_i = C(K, x_i) \sum_{j=1 \ and \ j \neq i}^{N} \frac{|| w_j || - || w ||_m}{r^n} \qquad (I)$$

where $C(K, x_i) \in C^1$. Thus, we obtain the new position of each grid point by integrating equation (I) for each point. For multi-dimensional spaces, we compute the velocities and the new positions of the grid points by integrating equations similar to (I) in each direction.

Next, we present the parallelization of the AR method in two dimensions on distributed memory multiprocessor systems that support a mesh interconnection topology. We employ systolic type techniques to eliminate synchronization delay, and minimize the communication overhead among processors. Our objectives in designing an algorithm for the parallel AR method are : *i*) the minimization of the so called edge/node contention (one or more links/nodes are shared between more than one paths in the computational graph of the algorithm, *ii*) minimization of the amount of data transferred between processors, and *iii*) minimization of the synchronization delay.

We assume that the rectangular computational space is decomposed into sub-blocks (or subdomains or tiles, see Figure 1) $D_{i,j}$ with $\frac{N}{\sqrt{P}} \times \frac{N}{\sqrt{P}}$ grid points per sub-block stored on each processor (i,j). The new position of the grid points in each $D_{i,j}$ sub-block is computed in $\sqrt{P}$ iterations. If we suppress the block indices, then the computation carried out by each processor in the kth iteration consists of (i) sending the magnitudes of the measure w of the grid points that the (i,j) processor received in the (k-1)th iteration to processors (i, (j-1)mod P) and ((i+1)mod P, j) respectively, (ii) computing the new positions using its local data, (iii) receiving the magnitudes of measure w from processors (i, (j+1) mod $\sqrt{P}$) and ((i-1) mod $\sqrt{P}$, j) respectively, and (iv) computing the new positions using the recently arrived data.

The interconnection of processor elements, which is a *folded grid* topology, and the distribution of input are shown in Figure 4.

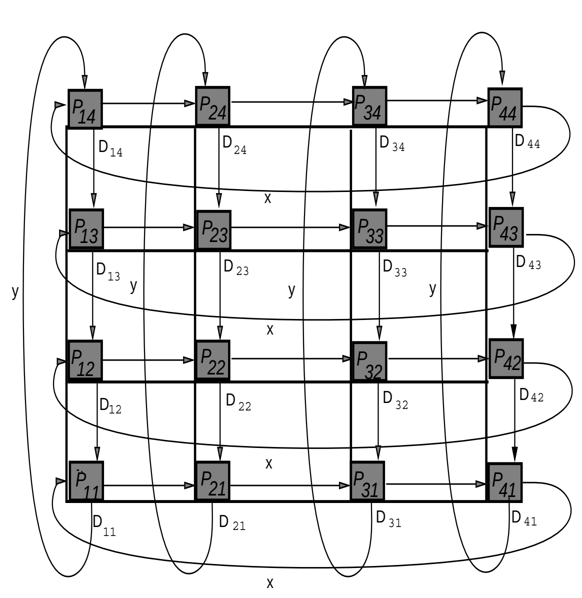


Figure 4: The interconnection network and the distribution of data. PE (i,j) computes the new position for the sub-block $D_{i,j}$. Data are moved along the x-path and y-path.

DISCUSSION AND CONCLUSION

In this paper, we described a methodology for developing a storm-scale weather prediction model, the Advanced Regional Prediction System (ARPS) developed by the Center for Analysis and Prediction of Storms, appropriate for use on massively parallel processors. Our approach is based on High Performance Fortran (HPF). A HPF implementation of ARPS will support immediate goals of model portability and scalability over distributed memory parallel architectures, shared memory vector architectures, and workstation-based distributed computing networks. The use of HPF as a base technology links ARPS development to industry-standard data parallel Fortran, while supporting future research plans such as implementing improved numerical methods, parallel algorithms, grid adaption techniques, and eventual operational demonstration. At the same time, we use the ARPS as a challenging application problem to help define future improvements to FortranD/HPF.

The CAPS research strategy is to continue developing the ARPS as the first cloud/meso-scale model designed specifically for parallel computers and for operational testing and possible implementation. The future availability of Teraflop computers, high bandwidth Doppler radars, and models such as the ARPS will make operational weather forecasting based on the ensemble strategy practical (e.g., Brooks et al. 1991). Ensemble forecasting involves the generation of multiple forecasts, valid for the same period, based on initial conditions that vary slightly from each other in a physically-plausible and statistically consistent manner. In this manner, one not only produces a forecast, but also information on the skill or variability of the forecast (e.g., Leith 1974; Kalnay 1987).

Linking parallel grid adaption methods to ARPS development and HPF is a key research issue. We will generate nested grids based on domain decomposition. Rectangular subdomains or tiles will be aligned locally with the the attraction/repulsion method. This method can be implemented using regular data structures, and existing directives of the current version of FortranD/HPF. Alternative approaches, such as continuous models based on partial differential equations require sparse data structures which are not yet available in FortranD/HPF. However, for the ARPS, continuous adaptation (Dietachmayer and Droegemeier 1992) is not deemed practical for two reasons: first, the global timestep is limited by the smallest zone in the mesh, and second, it is not possible to use different physics parameterizations with different resolutions, a feature which may be highly desirable in multi-scale flows, particularly with respect to subgrid-scale parameterizations.

ACKNOWLEDGEMENTS

This research was partially supported by the Center for Analysis and Prediction of Storms at the University of Oklahoma, which is sponsored by the National Science Foundation under Grant ATM88-09862; and the Northeast Parallel Architectures Center at Syracuse University, which is supported in part with equipment provided by the Center for Research on Parallel Computation with NSF cooperative agreement number CCR-8890615. Also, the first author gratefully acknowledges the Alex G. Nason Fundation for the Nason Prize Award that supports him at NPAC.

REFERENCES

Anderson D. A. and M.M. Rai. 1982. “The use of solution adaptive grids in solving partial differential equations." *In Numerical Grid Generation* J.F. Thompson, ed. Elsevier Science Publishing, Inc. 317-338.

Babuska, I. and W. C. Rheinboldt. 1978. "A posteriori error estimates for the finite element method." *Internat. J. Numer. Methods Engrg.* 12: 1957.

Berger, J.B. and J. Oliger. 1984. "Adaptive mesh refinement for hyperbolic partial differential equations." *Journal of Computational Physics.*, 53: 484-512.

Brackbill, J.U. and J.S. Saltzman. 1982. "Adaptive Zoning for Singular Problems in Two Dimensions." *Journal of Computational Physics.*, 46: 342-368.

Brooks, H.E., C.A. Doswell III, and R.A. Maddox, 1992: "On the use of mesoscale and cloud-scale models in operational forecasting. *Wea. Forecasting*, 7, 120-132.

CAPS. 1992. *ARPS Version 3.0 User's Guide.* Center for Analysis and Prediction of Storms, University of Oklahoma, 183 p.

Carpenter, R. Jr., K. Droegemeier, P. Woodward, and C. Hane. 1990. "Application of the piecewise parabolic method (PPM) to meteorological modeling." *Mon. Wea. Rev.*, 118: 586-612.

Chrisochoides, N.P., E.N. Houstis, S.B. Kim, M.K. Samartzis, and J.R. Rice. 1992, "Parallel Iterative Methods." To appear in the proceedings of the *7th IMACS International Conference on Computer Methods for PDEs*

Dietachmayer, G.S. and K.K. Droegemeier, 1992. "Application of Continuous Dynamic Grid Adaption Techniques to Meteorological Modeling. Part I: Basic Formulation and Accuracy". *Mon. Wea. Rev.*, 120, 1675-1706.

Droegemeier, K. 1990. "Toward a Science of Storm-Scale Prediction." In *Proceedings 16th Conference on Severe Local Storms* (Alberta Canada, Oct. 22-26). American Meteorological Society.

Droegemeier, K., K. Johnson, K. Mills, and M. O'Keefe. 1992. "Experiences with the scalable-parallel ARPS Cloud/Mesoscale Prediction Model on Massively Parallel and Workstation Cluster Architectures." In *Proceedings, 5th Workshop on the Use of Parallel Processors in Meteorology. European Center for Medium Range Weather Forecasts.* (Reading, England). Nov. 23-37.

Fiedler, B. and R.J. Trapp, 1993. "A Fast Dynamic Grid Adaption Scheme for Meteorological Flows." submitted to *Mon. Wea. Rev.*.

Gropp, W.D. and D. Keyes. 1990. "Semi-structured refinement and parallel domain decomposition methods." *In Unstructured scientific computation on scale multiprocessors* P. Mehrotra, J. Saltz and R. Voigt eds. 187-203.

Hageman, L.A. and D.M. Young. 1981. *Applied Iterative Methods.* Academic Press, 388p.

High Performance Fortran Forum. 1992. *DRAFT High Performance Fortran Language Specification.* Rice University. Houston, Texas. Version 0.4 Nov. 6, 1992. (available by anonymous ftp from minerva. npac.syr.edu, directory HPFF).

Kalnay, E. and A. Dalcher, 1987: Forecasting forecast skill. *Mon. Wea. Rev.*, 115, 349-356.

Leith, C.E., 1974:L "Theoretical skill of Monte Carlo forecasts". *Mon. Wea. Rev.*, 102, 409-418.

Lilly, D. 1990. "Numerical prediction of thunderstorms–has its time come?" *Quart. J. Roy. Meteor. Soc.*, 116:779-798.

McCormick, S. and D. Quinlan. 1989. "Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors : Performance Results" *Parallel Computing* 12: 145-156.

Paine, K.L., 1991. "A comparison of Two Methods for Dynamic Grid Adaption in Two-Dimensional Passive Scalar Transport." M.S Thesis, School of Meteorology, University of Oklahoma, 195p.

Paine, K.L. and K.K. Droegemeier. 1991. "A comparison of two methods for dynamic grid adaption in a two-dimensional scalar transport equation" *In Proceedings of Numerical Weather Prediction* (Denver, Colorado, Oct. 14-18), American Meteorological Society.

Rudin W. 1976. *Principles of Mathematical Analysis.* McGraw-Hill Book Company. 342p.

Russell, R.D. and J. Christiansen. 1978. "Adaptive mesh selection strategies for solving boundary value problems." *SIAM J. Numer. Anal.* 15: 59.

Staniforth, A. and J. Cote. 1991. "Semi-Lagrangian Integration Schemes for Atmospheric Models–A Review." *Monthly Weather Review*, vol. 119:2206- 2223.

Skamarock, W., 1989. "Truncation Error Estimates For Refinement Criteria In Nested And Adaptive Models". *Monthly Weather Review*, 117, 872-886.

Skamarock, W., J.J. Oliger, and R. Street, 1989. "Adaptive grid refinement for numerical weather prediction." *Journal of Computational Physics*, 80: 27-60.

Thompson J. F.., Z. U. A. Warsi, and C. W. Mastin, 1985. *Numerical Grid-Generation Foundations and Applications.* North-Holand, 483p.

Trease, H. and J. Cerutti. 1992. "The Free-Lagrange Method on the Connection Machine." In *Unstructured Scientific Computation on Scalable Multiprocessors.* P. Mehrotra,, J. Saltz, and R. Voigt, eds. The MIT Press, Cambridge, Massachusetts, 1-10.