Parallel Implementation of a Data-Transpose Technique for the Solution of Poisson's Equation in Cylindrical Coordinates [*]

Howard S. Cohl[†] Xian-He Sun [‡] Joel E. Tohline [§]

Abstract

We present a parallel finite-difference algorithm for the solution of the 3D cylindrical Poisson equation. The algorithm is based on a data-transpose technique, in which all computations are performed independently on each node, and all communications are restricted to global 3D data-transposition between nodes. The data-transpose technique aids us in implementing two sequential algorithms for the solution of the cylindrical Poisson equation. The first algorithm is based on the alternating direction implicit (ADI) method and the second is based on Fourier Analysis (FA). In both algorithms we first perform a Fourier transform in the naturally periodic azimuthal coordinate direction. This decouples the 3D problem into a set of independent 2D problems which are then solved by the ADI and FA methods. In the ADI method we convert each 2D problem into two sets of 1D (tridiagonal) problems which are then solved iteratively by alternating in the radial and vertical coordinate directions. In the FA method we convert each 2D problem into a set of tridiagonal problems by transforming into an orthogonal "Fourier" basis on whose type depends on the boundary conditions in the vertical coordinate direction. In both algorithms, the resulting tridiagonal systems are solved using LU decomposition with forward- and back-substitution. An operation and communication count analysis of these algorithms plus an internal timing analysis of the code is presented. Execution times are measured and compared on a 4K-node massively parallel MasPar MP-2 and on a single processor Cray C90 vector machine. Experimental results show that the new algorithm is efficient and provides good performance on the inexpensive MasPar machine due to it's high communication to computation ratio.

1 Introduction

With the advent of parallel computing, a great amount of research has been accomplished on the field of fast parallel algorithms for the numerical solution of partial differential equations (PDEs). In computational fluid dynamics (CFD) applications, the need for fast accurate solution of PDEs is usually required for large scale simulations. We focus our attention on Poisson's equation, which is a second order elliptic PDE.

We require the solution of Poisson's equation as a small part of a much larger fluid dynamics problem. We perform large-scale CFD simulations of massive astronomical fluid objects using a well-tested second-order accurate Eulerian fluid code in cylindrical

[*] This research was supported in part by the National Aeronautics and Space Administration under NASA grant NAGW-2447, NAS1-1672/MMP, by the National Science Foundation grant AST-9008166, and by the DOE (AGTSR) program through subcontract 93-01-SR015.

[†]Dept. of Physics and Astronomy, Louisiana State University, Baton Rouge, LA

[‡]Dept. of Computer Science, Louisiana State University, Baton Rouge, LA

[§]Dept. of Physics and Astronomy, Louisiana State University, Baton Rouge, LA coordinates [1]. In our application, we require the solution of Poisson's equation in order to completely describe the gravitational forces exerted by a massive astronomical fluid object on its surroundings and on itself.

We adopt a finite-difference method for the solution of the cylindrical Poisson equation. In iterative schemes, the solution of the PDE is obtained by starting with an initial guess and then proceeding iteratively until the solution is obtained to within desired accuracy. In direct schemes, the solution of the PDE is obtained by direct numerical solution of the finite-difference equations. Direct methods are usually preferred over iterative methods since they are guaranteed to generate an exact solution.

In this paper we utilize both a fast direct method and an iterative method to solve the problem at hand. Both methods are implemented in parallel using a data-transpose technique. The data-transpose technique is a parallelization strategy in which all communication is restricted to global 3D data-transposition operations and all computations are subsequently performed with perfect load balance and zero communication.

We present an implementation of this technique on the MasPar MP-2, which is a massively parallel, distributed memory, single instruction stream multiple data stream (SIMD) architecture. We test our algorithm and study its performance on the MasPar MP-2 and compare timings with a fully-vectorized version of the code on a Cray C90 vector processor.

In the remainder of the paper, we present a detailed description of our parallel algorithm. In section 2, we describe the sequential algorithms that we use in conjunction with the data-transpose technique in order to solve the cylindrical Poisson equation. In section 3, we describe the parallel data-transpose technique, how it applies to the two sequential algorithms we presented in section 2 and a theoretical description of how the technique is applied on a 2D mesh topology. In section 4, we present the parallel implementation of the technique on the MasPar MP-2 architecture and discuss timings of our code. Finally, in section 5, we summarize our conclusions.

2 Sequential Algorithms

We present a parallel method to solve Poisson's equation

$$\nabla^2 \Phi = f, \tag{1}$$

where Φ is the potential, f is the source function, and ∇^2 denotes the Laplacian operator.

In a cylindrical geometry, the Cartesian vertical coordinate, z, remains unchanged, but the Cartesian x and y coordinates are replaced by the polar coordinates r (radial) and θ (azimuthal) via the transformation

$$\begin{aligned} x &= r\cos\theta, \\ y &= r\sin\theta. \end{aligned} \tag{2}$$

Our domain boundaries are specified by the conditions

$$\begin{aligned} 0 \le\ & r \le R, \\ 0 \le\ & \theta \le 2\pi, \\ & |z| \le Z, \end{aligned} \tag{3}$$

where R is the radius of the cylindrical domain, $2Z$ is the height of the cylinder, and the angle θ is measured in radians

We perform a spatial discretization of our domain using the indices (j,k,l) to refer to the (r_j, z_k, θ_l) positions of each cell center with $\Delta r_j, \Delta z_k$, and $\Delta\theta_l$ being the radial, vertical and azimuthal grid spacing of each cell. Both the potential and the source function are evaluated at the center of each grid cell. The discretization is performed using J radial zones, K vertical zones, and L azimuthal zones.

In our applications, mixed Dirichlet, Neumann, and periodic boundary conditions are usually assumed for the potential Φ, viz.

$$\begin{aligned} \Phi(R,\theta,z) &= g(\theta,z), \\ \Phi(r,\theta,+Z) &= h_+(r,\theta), \\ \Phi(r,\theta,-Z) &= h_-(r,\theta), \\ \Phi(r,2\pi,z) &= \Phi(r,0,z), \end{aligned} \tag{4}$$

where the functions g, h_+, and h_- are known along the corresponding boundaries. The interior of our domain is mapped onto a rectangular computational grid which extends between $2 \le j \le J+1$, $2 \le k \le K+1$, and $1 \le l \le L$. Since the azimuthal coordinate direction is periodic in nature, the index l runs modulo L, i.e. $l + L = l$. We also require continuity of the solution across the z axis, this is evaluated at $j = 1$ by setting $\Phi(1,k,l) = \Phi(2,k,l+L/2)$. The boundary values g, h_- and h_+ are evaluated at $j = J+2, k = 1$ and $k = K+2$, respectively.

2.1 Finite-Difference Derivation of 2D Helmholtz Equation

Written in cylindrical coordinates Poisson's equation reads

$$\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial\Phi}{\partial r}) + \frac{1}{r^2}\frac{\partial^2\Phi}{\partial\theta^2} + \frac{\partial^2\Phi}{\partial z^2} = f(r,\theta,z). \tag{5}$$

The finite-difference representation of eq. (5) is

$$\frac{1}{r_j\Delta r_j}\Big(r_{j+1/2}\frac{\Delta_{j+1/2}\Phi}{\Delta r_{j+1/2}} - r_{j-1/2}\frac{\Delta_{j-1/2}\Phi}{\Delta r_{j-1/2}}\Big) + \frac{1}{r_j^2}\frac{\Delta_l^2\Phi}{(\Delta\theta_l)^2} + \frac{\Delta_k^2\Phi}{(\Delta z_k)^2} = f_{j,k,l}, \tag{6}$$

where, for every index i, the symbols Δ_i and Δ_i^2 indicate the sense that the first and second differences are taken and their proper centering. For example, $\Delta_{j+1/2}$ denotes that the first difference is taken in the radial coordinate direction and is centered at the $(j+1/2,k,l)$ location, while Δ_l^2 denotes that the second difference is taken in the azimuthal coordinate direction and is centered at the (j,k,l) location. Similarly $\Delta r_j = r_{j+1/2} - r_{j-1/2}$, $\Delta r_{j+1/2} = r_{j+1} - r_j$, $\Delta\theta_l = \theta_{l+1/2} - \theta_{l-1/2}$, and $\Delta z_k = z_{k+1/2} - z_{k-1/2}$. Expanding the differences Δ and Δ^2 up to second order in the grid spacings $\Delta r_j, \Delta z_k$, and $\Delta\theta_l$, one obtains

$$\begin{aligned} &A(j)\ \Phi(j+1,k,l) + B(j)\ \Phi(j-1,k,l) \\ &+ C(j,l)\ \Big[\Phi(j,k,l+1) + \Phi(j,k,l-1)\Big] + D(k)\ \Big[\Phi(j,k+1,l) + \Phi(j,k-1,l)\Big] \\ &-\Big\{A(j) + B(j) + 2[C(j,l) + D(k)]\Big\}\ \Phi(j,k,l) = f(j,k,l). \end{aligned} \tag{7}$$

The coefficients in eq. (7) are defined by the expressions

$$\begin{aligned} A(j) &= r_{j+1/2}(r_j\Delta r_{j+1/2}\Delta r_j)^{-1}, \\ B(j) &= r_{j-1/2}(r_j\Delta r_{j-1/2}\Delta r_j)^{-1}, \\ C(j,l) &= (r_j\Delta\theta_l)^{-2}, \\ D(k) &= (\Delta z_k)^{-2}. \end{aligned} \tag{8}$$

The 3D problem represented by eq. (7) can be decoupled into a set of independent 2D problems by performing a discrete Fourier transform in the azimuthal coordinate direction of the general form

$$Q(j,k,l) = \sum_{m=0}^{L/2} \Big\{ Q_m^1(j,k)\cos(m\theta_l) + Q_m^2(j,k)\sin(m\theta_l) \Big\}, \tag{9}$$

where Q denotes either Φ or f, and Q_m^i, with $i=1$ and $i=2$, are the Fourier coefficients of the cosine and the sine terms, respectively. Substituting eq. (9) into eq. (7), assuming a constant value of $\Delta\theta_l \equiv \Delta\theta = 2\pi/L$, and accounting for the continuity boundary condition across the z-axis, one obtains

$$\begin{aligned} &A(j)\ \phi_m^i(j+1,k) + B(j)\ \Big\{1+\delta_{j2}[(-1)^m - 1]\Big\}\phi_m^i(j-1+\delta_{j2},k) \\ &\qquad +D(k)\ \Big[\phi_m^i(j,k+1) + \phi_m^i(j,k-1)\Big] \\ &+\Big\{2(\lambda_m - 1)C(j) - [A(j)+B(j)+2D(k)]\Big\}\ \phi_m^i(j,k) = f_m^i(j,k), \end{aligned} \tag{10}$$

where δ is the Kronecker symbol,

$$\begin{aligned} m &= 0,1,2,\ldots,L/2 \qquad (\text{ for } \ i=1\ ), \\ m &= 1,2,3,\ldots,L/2-1 \qquad (\text{ for } \ i=2\ ), \end{aligned} \tag{11}$$

and $\lambda_m \equiv \cos(m\Delta\theta)$. The equations for $i=1$ are derived by equating coefficients of the cosine terms in the Fourier expansion and the equations for $i=2$ are derived by equating coefficients of the sine terms. Eq. (10) is the finite-difference representation of a 2D Helmholtz Equation.

2.2 Solution Methods for Each 2D Helmholtz Equation

In this section we present the sequential methods that we use to obtain the solution for each 2D Helmholtz equation.

2.2.1 The ADI Method The alternating direction implicit (ADI) [3, 2] method is a widely used iterative method for solving multi-dimensional boundary value problems. It is an operator-splitting scheme which solves implicitly, and in an alternating fashion, each of the dimensions of a multi-dimensional elliptic problem. It combines two ideas, described below, and results in a rapidly convergent and numerically stable algorithm. (See also [4] and [5] and references given therein for implementations of the same technique to the solution of various PDEs.)

The first idea of ADI is to write the original operator equation in the form of a diffusion equation, viz.

$$\frac{\partial \Phi}{\partial t} = \nabla^2\Phi - f. \tag{12}$$

The diffusion equation uses a false dimensionless time which helps the algorithm settle into a final steady-state solution of eq. (1). We have adopted the prescription proposed by Black and Bodenheimer [5] to compute the "time steps" for a variable number of iterations. The second idea incorporated in the ADI technique is to implement a partially implicit solution of the 2D finite-difference equations. This is performed by splitting the terms of the 2D equations in such a way that, at each step, the finite-difference terms in two given directions are treated as known and unknown, respectively. When this is done, each 2D equation transforms into a set of tridiagonal equations. We then use the optimal sequential tridiagonal solver, LU decomposition with forward- and back-substitution (hereafter referred to as just LU decomposition) to solve each tridiagonal matrix. [4]

In our specific implementation, the spatial operator in the 2D Helmholtz equation (10) along with the diffusion term in eq. (12) is split as follows. For each choice of the Fourier mode elements i and m, during the r-sweep we use

$$\begin{aligned}(13)\qquad & A(j)\ \phi^{n+1/2}(j+1,k)+B(j)\Big\{1+\delta_{j2}[(-1)^m-1]\Big\}\phi^{n+1/2}(j-1+\delta_{j2},k)\\ & -\Big[A(j)+B(j)-2(\lambda-1)C(j)+\tfrac{2}{\Delta t}\Big]\phi^{n+1/2}(j,k)=f^n(j,k)\\ & -D(k)\ \Big[\phi^n(j,k+1)+\phi^n(j,k-1)\Big]+(2D(k)-\tfrac{2}{\Delta t})\ \phi^n(j,k);\end{aligned}$$

and during the z-sweep we use

$$\begin{aligned}(14)\qquad & D(k)\ \Big[\phi^{n+1}(j,k+1)+\phi^{n+1}(j,k-1)\Big]-(2D(k)+\tfrac{2}{\Delta t})\phi^{n+1}(j,k)=f^{n+1/2}(j,k)\\ & -A(j)\ \phi^{n+1/2}(j+1,k)-B(j)\ \Big\{1+\delta_{j2}[(-1)^m-1]\Big\}\phi^{n+1/2}(j-1+\delta_{j2},k)\\ & -\Big[2(\lambda-1)C(j)-(A(j)+B(j)-\tfrac{2}{\Delta t})\Big]\ \phi^{n+1/2}(j,k).\end{aligned}$$

In both cases, as indicated by the "time step" superscripts n, $n+1/2$, or $n+1$, terms on the right-hand side of the expressions are considered known quantities, and terms on the left are considered unknown.

2.2.2 Fourier Analysis One popular method for the solution of Poisson's equation is to use Fourier Analysis [6] in order to convert the 3D problem into a set of completely decoupled 1D problems. This method is highly efficient and takes advantage of the fast Fourier transform (FFT) algorithm. The resulting tridiagonal system can then be solved directly using LU decomposition.

As was mentioned earlier, we usually assume Dirichlet-Dirichlet (DD) boundary conditions in the vertical coordinate direction. This boundary condition can be accommodated by applying a *sine* transform in the vertical coordinate direction. In fact, any combination of Neumann and Dirichlet boundary conditions can be dealt with by using the appropriate discrete transform. [7, 8, 9] For instance, in the case of Neumann-Dirichlet (NN) boundary conditions, a discrete quarter *cosine* transform would accomplish the decoupling. All of the possible combinations can be obtained using certain appropriate pre- and post-conditioning operations on the input and output of a standard FFT routine. [9]

When the appropriate transform is substituted in eq. (10) and uniform zoning is utilized (i.e. $\Delta z_k \equiv \Delta z = constant$), one obtains

$$\begin{aligned}(15)\qquad & A(j)\ \phi^i_{m,k'}(j+1)+B(j)\ \Big\{1+\delta_{j2}[(-1)^m-1]\Big\}\phi^i_{m,k'}(j-1+\delta_{j2})\\ & +\Big\{2(\lambda_m-1)C(j)+2(\lambda_{k'}-1)D-[A(j)+B(j)]\Big\}\ \phi^i_{m,k'}(j)=f^i_{m,k'}(j),\end{aligned}$$

where $\lambda_{k'}$ depends on the specific Fourier basis transform. Once the solution to this tridiagonal system is obtained via LU decomposition, the appropriate inverse transform is then applied in the vertical coordinate direction followed by an inverse Fourier transform in the azimuthal coordinate direction in order to bring the solution back into coordinate space.

3 Parallel Data-Transpose Technique

A large effort has gone into the development of fast sequential algorithms for the solution of Poisson's equation (see [4] for many examples). On shared-memory architectures, the sequential algorithms with the lowest operation count are optimal given a way to distribute the computations uniformly among the processors. [10] Similarly, a large effort has gone into the development of fast parallel algorithms for the solution of Poisson's equation on distributed-memory architectures (see [11, 12, 13]). We adopt a strategy for the parallel solution of the cylindrical Poisson equation on a distributed-memory architecture that has perfect computational load balance.

If a sequential algorithm requires a recursive sweep in one coordinate direction, then this sweep can be performed at each of the other coordinate locations independently. If we distribute our data in such a way that the coordinate direction in which the sweep needs to be performed is stored in the internal memory of each processor (node), then the computation can be performed in parallel on each of the nodes. Since each recursive sweep is performed completely in memory, no communication is required in the part of the calculation. The next sweep that needs to be performed is distributed across a set of processors. If we perform this recursive sweep with no change in the data distribution, then inter-processor communication will be required in order to perform the sweep, with the recursive nature of the sweep leading to poor load balance. Another choice that we have is to perform a global data-transposition operation on the storage array so that the second sweep direction is redistributed into the internal memory of each node. The question of which choice is optimal is architecture dependent.

On a 2D mesh of processors, the most natural way to map a 3D array is to spread two of the array directions out across the processors (X and Y processor grid directions) and to store the third array direction in the internal memory of each node (M). If we perform the data-transposition operation between the X processor grid direction and M, then the global data-transpose can be performed in parallel for each Y processor grid location. Similarly, if we perform the data-transposition operation between the Y processor grid direction and M, then the global data-transpose operation be be performed in parallel for each X processor grid location. [14] We perform a data-transposition operation between each computational sweep for the preceding two algorithms.

4 Analysis

4.1 Specific Implementation - The MasPar Architecture

Our Poisson solver was developed on an 4,096-node (64×64 grid of processors) MasPar MP-2. The MasPar MP-2 is a distributed-memory, SIMD computer in which a 2D (X, Y) grid of processor elements (PEs) comprises the primary compute engine of the machine. This machine is configured such that each PE contains 128 K-Bytes of local dynamic RAM; hence, the 4K-node system contains 0.5 G-Bytes. The basic topology of the MP-2 is that of a 2D torus with a communication network designed such that each of the PEs communicates most efficiently with its eight nearest neighbors via X-net hardware and long range communication can be handled via a global router network. [15].

4.2 Theoretical Timing Analysis

Since the data-transpose technique allows both sequential algorithms to be implemented, in parallel, with no change in the computation strategy, the sequential operation counts gives us a partial measure of the parallel execution time. A more accurate model for the total

grid size	MP-2, FA	MP-2, ADI	C90, FA	C90, ADI
8^3	95.5	106	1.35	2.34
16^3	131	166	5.10	12.3
32^3	219	290	29.6	94.3
64^3	365	504	237	752

TABLE 1
Comparison : Execution Times in milliseconds

parallel execution time must include the data-transpose operations in the analysis. In both algorithms, a forward and inverse FFT is applied in the azimuthal coordinate direction. Including the highest order terms, the sequential operation count for a length p real FFT is $\frac{1}{2}(5p\log_2(p) + 13p)$. [8] Similarly, the operation count for a length p tridiagonal solution using LU decomposition is $5p$. [4] The sequential operation count for the real fast *sine* transform is $\frac{1}{2}(5p\log_2(p) + 22p)$. [9] In order to simplify the calculation, we assume equal number of grid points in all three coordinate directions, i.e. $N = J = K = L$. Therefore, the total sequential operation count for ADI is $N^2[5N\log_2(N) + 13N + 10NI]$, where I is the total number of iterations needed to converge to a solution. As an example, the total sequential operation count for Fourier Analysis applied to the fast *sine* transform method for the Dirichlet-Dirichlet boundary condition is $N^2[10N\log_2(N) + 40N]$.

Given these sequential operation counts we can then estimate the parallel execution time. In the ADI algorithm, the data-transpose function is called $3 + 2I$ times and in the Fourier analysis algorithm it is called 4 times. In the case where each data-element is mapped to a single processor, the parallel operation count will be equal to the sequential operation count, reduced by a factor of N^2. Including this fact and including the amount of time spent in the data-transpose function, we estimate the parallel execution time for the ADI algorithm to be

$$t_{ADI} = [5N\log_2(N) + 13N + 10NI]C + (3 + 2I)t_{TRAN}, \tag{16}$$

and we estimate the parallel execution time for the Fourier analysis algorithm to be

$$t_{FA} = [10N\log_2(N) + 40N]C + 4t_{TRAN}, \tag{17}$$

where C is a constant that determines how much time is spent on average per operation count and t_{TRAN} is the amount of time that is takes to complete a single data-transpose operation.

4.2.1 Timings The timings of the new parallel algorithm are given in Table 1. In this table we list the size of the computational grid and the corresponding timings in milliseconds on the MasPar MP-2 and Cray C90 for the implementation of the ADI and FA algorithms. We can see from the data that when the entire computational grid is used (64^3) on the MasPar MP-2, the execution times are competitive with that obtained on the Cray C90. This fact is a demonstration of the highly parallel nature of our algorithms.

5 Conclusion

We present a parallel implementation of a data-transpose technique for the finite-difference solution of the 3D cylindrical Poisson equation. We implement two algorithms in this technique, ADI and Fourier analysis. The ADI algorithm is iterative and can handle non-uniformly zoned radial and vertical meshes. The Fourier analysis algorithms is direct and requires uniform vertical grid spacing. We implement these algorithms on a MasPar MP-2 and demonstrate the highly parallel nature of our solution strategy.

Acknowledgments

The authors would like to thank Sandeep Dani for his invaluable assistance in the development of the parallel algorithms and the San Diego Supercomputer Center for the usage of the Cray C90. We also acknowledge useful discussions that we have had with S. Acharya and G. Pointel on this topic. We also acknowledge the Scalable Computing Lab at Iowa State University for the use of their MasPar MP-2. We acknowledge the support that has been received from the Louisiana State Board of Regents through the Louisiana Education Quality Support Fund for the establishment of the Concurrent Computing Laboratory on whose facilities this research was partially conducted.

References

[1] J. Woodward, J. Tohline, and I. Hachisu, "The Stability of Thick, Self-Gravitating Disks in Protostellar Systems," *Astrophys. J.*, vol. 420, p. 274, 1994.

[2] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations.* Wadsworth & Brooks/Cole, Mathematics Series, 1989.

[3] D. Peaceman and J. H.H. Rachford *J. Soc. Indust. Appl. Math.*, vol. 3, p. 28, 1955.

[4] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in FORTRAN, The Art of Scientific Computing.* Cambridge University Press, 1992. Second Edition.

[5] D. Black and P. Bodenheimer *Astrophys. J.*, vol. 199, p. 619, 1975.

[6] R. Hockney, "A Fast Direct Solution of Poisson's equation using Fourier analysis," *J. ACM*, vol. 12, pp. 95–113, 1965.

[7] P. Swarztrauber, "Direct Method for Discrete Solution of Seperable Elliptic Equations.," *SIAM, J. Numer. Anal.*, vol. 11, p. 1136, 1974.

[8] P. Swarztrauber, "The Methods of Cyclic reduction, Fourier analysis and the FACR Algorithm for the Discrete Solution of Poisson's Equation on a Rectangle," *SIAM Review*, vol. 19, p. 490, 1977.

[9] P. L. J.W. Cooley and P. Welch, "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine and Laplace Transforms," *J. Sounds Vib.*, vol. 12, p. 315, 1970.

[10] W. Briggs and T. Turnbull, "Fast Poisson Solvers for MIMD computers," *Parallel Computing*, vol. 6, p. 265, 1988.

[11] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing, Design and Analysis of Algorithms.* The Benjamin/Cummings Publishing Company, Inc., 1994.

[12] R. S. et. al., "FFTs and Three-Dimensional Poisson Solvers for Hypercubes," *Parallel Computing*, vol. 17, p. 121, 1991.

[13] U. Schwardmann, "Parallelization of a Multigrid solver on the KSR1," *Supercomputer*, vol. 55, p. 4, 1993.

[14] J. D. J Choi and D. Walker, "Parallel Matrix Transpose Algorithms on Distributed memory concurrent computers," *Parallel Computing*, vol. 21, p. 1387, 1995.

[15] X.-H. Sun and D. Rover, "Scalability of Parallel Algorithm-Machine Combinations," *IEEE Transactions on Parallel and Distributed Systems*, pp. 599–613, June 1994.