1996

Parallel Reuse Methodologies for Elliptic Boundary Value Problems

S. Markus

Elias N. Houstis
Purdue University, enh@cs.purdue.edu

Report Number:
96-056

PARALLEL REUSE METHODOLOGIES FOR ELLIPTIC BOUNDARY VALUE PROBLEMS

Shahani Markus
Elias N. Houstis

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR 96-056
September 1996

Parallel Reuse Methodologies for Elliptic Boundary Value Problems

S. Markus and E. N. Houstis
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA.

March 1, 1996

Abstract

We describe two parallel frameworks that allow the *reuse* of the discretization part of sequential general elliptic PDE (partial differential equation) solvers. These parallel reuse methodologies are based on the "divide and conquer" computational paradigm. They have been integrated into the Parallel ELLPACK problem solving environment that supports PDE computing across many hardware platforms. Experimental results indicate the effectiveness of the reuse frameworks implemented.

We also evaluate the performance of the Parallel ITPACK library of stationary iterative solvers. This package has been implemented using several message passing communication libraries. We consider the parallel solution of sparse algebraic equations obtained from the discretization of second order elliptic PDEs using finite difference and finite element techniques. The performance of the Parallel ITPACK solvers is measured on many distributed memory platforms including clusters of workstations.

1 Introduction

Computational models based on partial differential equation (PDE) mathematical models have been successfully applied over the last 50 years to study many physical phenomena and design a variety of artifacts. The overall quantitative and qualitative accuracy of these computational models in representing the physical situations or artifacts that they are supposed to simulate, depends very much on the computer resources available. The recent advances in high performance computing technologies have provided an opportunity to significantly speed up these computational models and dramatically increase their numerical resolution and complexity. However, there is significant state–of–the–art "legacy" software for the numerical solution of PDE models. It represents hundreds of man years of effort, and it would be unrealistic to expect these to be transformed by hand (in the absence of parallelizing compilers) on some virtual or physical parallel environment. The purpose of this paper is to formulate parallel methodologies that are capable of *reusing* parts of the sequential PDE solvers. For simplicity of this exposition we focus on computational models derived from elliptic PDE models and implemented in the Parallel ELLPACK PSE (problem solving environment) [16].

Most of the parallelization techniques presented here are applicable to general semi–discrete and steady–state models. Specifically, we consider PDE models consisting of a PDE equation ($\mathbf{Lu} = \mathbf{f}$), defined on some region (Ω) and subject to some auxiliary condition ($\mathbf{Bu} = \mathbf{g}$) on the boundary ($\partial\Omega$). It appears that one can formulate many (thousands) computational models to simulate the above general mathematical model, depending on the approximation technique selected to discretize the domain of definition, the PDE equation and boundary conditions, or the PDE approximate solution. In this article, we consider parallel computational models based on the popular discretization technique of piecewise–linear polynomial (finite element) approximation of the solution $\mathbf{u}$. In the parallel computational models considered, the continuous PDE problem is reduced to a distributed sparse system of linear equations. Depending on the type of the PDE operators $\mathbf{L}$ and $\mathbf{B}$ and the simplicity/regularity of the PDE region, the corresponding finite element system of equations can be solved by general or rapid parallel algebraic solvers. The following discussion is focused on parallelization techniques that allow both to reuse existing ("legacy") PDE software parts and provide a *template* or *framework* to build new parallel PDE software.

2 Parallelization Methodologies for "Legacy" Elliptic PDE Software

The "legacy" software can be classified in two large classes. The first class contains customized PDE software for specific applications. This software is usually difficult to be adapted for the simulation of an alternative application. The second class contains PDE software that supports the numerical solution of well defined mathematical models which can be used easily to support the simulation of multiple applications. The first class tends to be application domain specific, thus the parallelization efforts and results appear in many diverse sources. In this article we consider parallelization techniques for the second class of PDE software. Table 1 lists some of the public domain "legacy" software that is available in the parallel ELLPACK PSE.

The majority of the code of each PDE system is implementing the geometric and the PDE model discretization phases. This tends to be the most knowledge intensive part of the code. The rest of the code deals with the solution of the discrete finite difference or finite element equations. This phase is well understood and many alternative solution paths exist. In Table 2 we summarize the above observations and in its last column we estimate the parallelization effort needed to convert or

Name	*Reference*
ELLPACK	[27]
//ELLPACK	[14, 15]
FIDISOL	[29]
VECFEM	[11]
CADSOL	[28]
PDEONE	[32]
PDECOL	[21]
PDETWO	[31]
MGGHAT	[24]

Table 1: *PDEpack: Public domain "legacy" PDE software*

re-implement the components of the "legacy" PDE code into some parallel environment "by hand". From this analysis, it is clear that any parallel methodology that attempts to reuse the PDE discretization software parts is well justified. In the following, we describe three parallel methodologies that are based on some "optimal" non–overlapping partitioning of the discrete PDE geometric data structures (i.e. meshes). Figure 1 depicts these three decompositions approaches for a two dimensional region and the message passing computational paradigm. The two left most paths in Figure 1 depict methodologies that support the reuse requirement. The third path provides a framework to develop new customized parallel code for the discretization part of the PDE computation. All three approaches assume the availability of parallel linear solvers implemented on distributed algebraic data structures obtained through an "optimal" non–overlapping partitioning of the corresponding PDE geometric data structures.

3 An Off–Line Reuse Parallel Methodology for "Legacy" PDE Software

Figure 1a depicts an off-line approach, referred to as M^+, which assumes that the discretization of the PDE model is realized by an existing sequential "legacy" PDE code, while it goes off-line to a

Components	*Computational Intensity*	*Knowledge Intensity*	*Parallelization Effort*
Geometric discretization	$O(N)$	Very High	Significant
PDE model discretization	$O(N)$	Very high	Significant
Solution	$O(N^\alpha), 1 < \alpha \leq 3$	Well understood, High	Relatively easy
Solution visualization	$O(N)$	High	Special hardware

Table 2: *The complexity of the elliptic PDE software parts and an estimate of the parallelization effort needed to implement them in some parallel environment, where N denotes the size of the discrete problem*

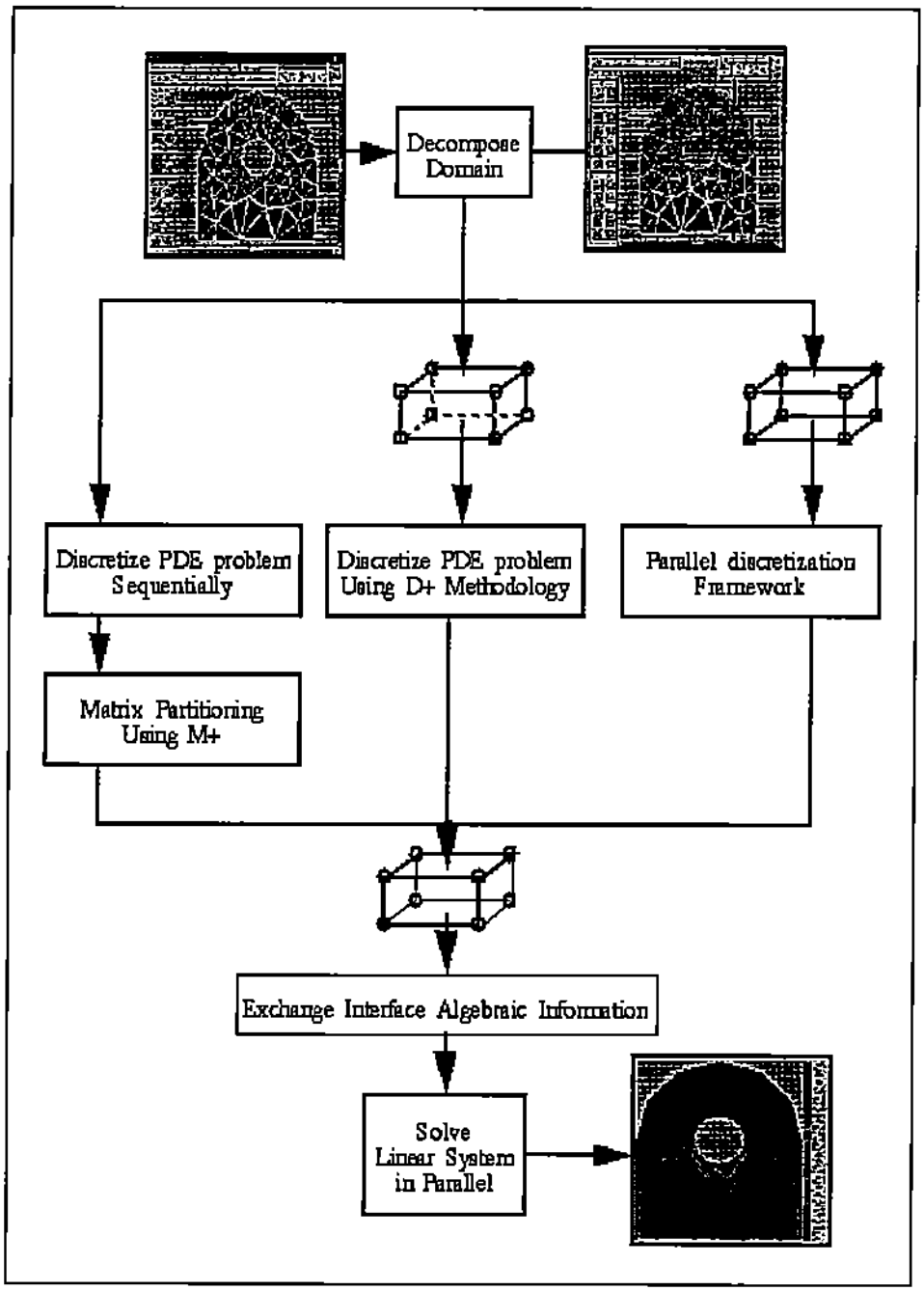

Figure 1: Three domain decomposition based parallel methodologies for elliptic PDEs. The left path (a) depicts an off-line parallel approach for solving the sequentially generated PDE equations, the center path (b) depicts an on-line non-overlapping domain decomposition approach capable of reusing existing discretization PDE software, and the right path (c) depicts a framework for developing new parallel PDE software.

parallel machine to solve the system of discrete equations. For the parallel solution of the discrete PDE equations, a decomposition of the sequentially produced algebraic system is required. It can be either *implicitly* obtained through a decomposition of the mesh or grid data or *explicitly* specified by the user. Then, the partitioned system is down–loaded onto the parallel machine. This is the most widely used methodology, since it allows for the preservation of the most knowledge intensive part of the code and for speeding up the most computationally intensive one. The obvious disadvantage of this approach is the memory bottleneck of the sequential server.

3.1 M^+ Tool Implementation

The current version of the parallel ELLPACK PSE includes a software tool to support the M^+ reuse methodology for the "legacy" software listed in Table 1. This self–contained, interactive, graphical tool can be used to obtain a domain decomposition–based partition of the discrete algebraic equations generated by any PDE software. Input to the tool consists of the linear system and a corresponding non–overlapping domain decomposition (figure 2).

In addition to the linear system partitioning module, the M^+ tool also contains modules for linear system visualization, parallel solver specification and template–based parallel driver program generation. The linear system input is represented in a tagged, self–identifying format and the domain decomposition input is accepted in the current parallel ELLPACK decomposition file format. The visualization module supports the display of the original and partitioned linear systems and also has a facility to permute a partitioned system to view it in arrowhead format. Based on the parallel solver specification, the parallel solution module generates the parallel solver driver program from a template and the requisite input data files for the node processors of the MIMD platform. The solution module also provides a facility for the compilation and execution of the parallel solver on the target hardware platform.

4 A Parallel Framework for Building New PDE Software

Figure 1c corresponds to a framework for developing customized PDE software. It is defined by a set of pre-defined decomposed geometric and algebraic data structures and their interfaces. The decomposition of the PDE data structures is chosen so that the underlying computations are uniformly distributed among processors and the interface length is minimum. This parallel framework has been used by many researchers to implement PDE based applications [6], [22], [26], [30] [33], [34]. Also, this framework has been used for developing general PDE software [16]. The parallel PDE solvers implemented on the above framework are distinguished primarily by the way they handle the interface equations and unknowns. An overview of the various parallel solution strategies proposed for handling the interface and interior equations can be found in [5].The simplest of these parallel strategies calls for the implementation of efficient sequential algebraic solvers on the framework data structures through the use of parallel sparse BLAS [3] that employ message passing primitives to exchange or accumulate interface quantities and carry out matrix-vector and vector-vector operations. The advantage of this approach is the fact that no new theory is required. Such parallel PDE solvers based on certain instances of finite difference and finite element schemes for elliptic PDEs can be found in the parallel ELLPACK PSE [15], [16]. These PDE solvers are described in [19] together with their performance evaluation. This study indicates that they can deliver significant speedups even for moderate size problems.

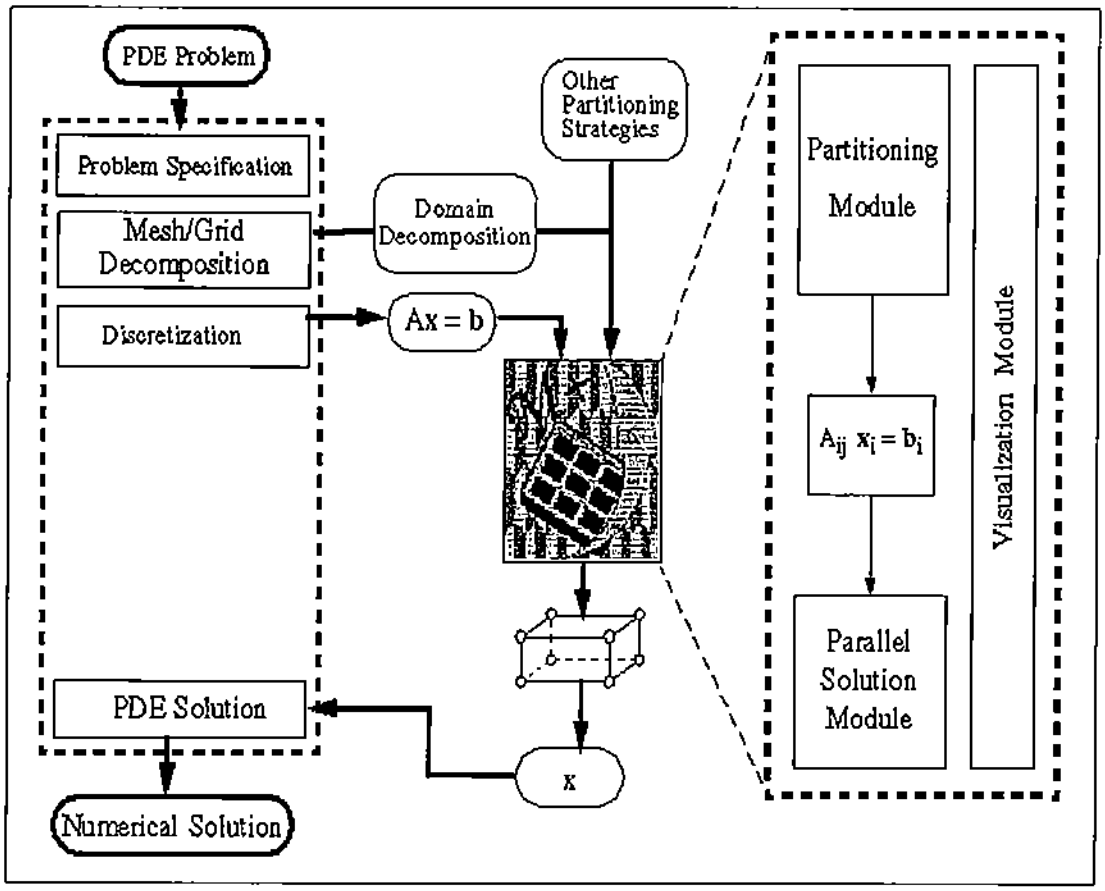

Figure 2: *An illustration of the M^+ methodology and components of the M^+ tool*

5 An On-Line Reuse Parallel Methodology for "Legacy" PDE Software

In Figure 1b we illustrate a third methodology for developing parallel PDE software that supports the reuse of existing PDE codes and attempts to address the shortcomings of the previous two methods. It is referred to as the D^+ methodology. The basic idea of this approach is to use the mesh decomposition to define a number of auxiliary PDE problems that can be discretized independently using the "legacy" PDE codes. Depending on the PDE operator and the approximation scheme used, appropriate continuous *interface conditions* must be selected to guarantee that the parallel on-line generated system of equations is complete and equivalent (apart from round-off error) to the sequential discrete algebraic system. These auxiliary linear systems are viewed as distributed components of a single system of linear equations and solved using a parallel linear system solver. A software environment that supports the D^+ approach for elliptic PDEs is available in the parallel ELLPACK PSE.

5.1 Mathematical Formulation of D^+ for the Finite Element Method

For second–order, self–adjoint elliptic PDEs of the form

$$Lu = -(\alpha u_x)_x - (\alpha u_y)_y + \beta u = \gamma,$$

we consider the formulation of the Ritz–Galerkin finite element method (FEM) based on bi–linear piecewise polynomials on a triangular finite element mesh of a domain $\Omega \subset \mathbf{R}^2$, with boundary conditions,

$$u = \rho \quad \text{on } \partial\Omega_1$$

$$u_n + \sigma u = \xi \quad \text{on } \partial\Omega_2 = \partial\Omega - \partial\Omega_1.$$

Then for an element E_l, we obtain the variational formulation

$$\int\int_{E_l} (\alpha u_x v_x + \alpha u_y v_y + \beta u) dx dy + \int_{(\partial\Omega_2 \cap E_l)} (\alpha\sigma uv) ds = \int\int_{E_l} (\gamma v) dx dy + \int_{(\partial\Omega_2 \cap E_l)} (\alpha\xi v) ds$$

for all test functions v in an appropriate Hilbert space.

We assume that the elements between the nodal interfaces of neighboring subdomains of the non–overlapping finite element mesh decomposition consists of one layer. Each subdomain along with its single layer of interface elements is regarded as the domain of an auxiliary PDE problem. Figure 3 illustrates a typical non–overlapping mesh decomposition for a PDE problem and the corresponding discrete domains of the auxiliary PDE problems.

We assign the Neumann condition $u_n = 0$ along the pseudo–boundaries of the auxiliary PDE problem domains. This results in an element stiffness matrix computation of the form,

$$\int\int_{E_l} (\alpha \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \alpha \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} + \beta \phi_i) dx dy = \int\int_{E_l} (\gamma \phi_j) dx dy$$

for elements E_l along the pseudo–boundaries of these auxiliary PDE problems.

These element stiffness matrix computations are identical to the computations that occur on the interior mesh elements of the original PDE problem. Hence combining the linear systems resulting from the finite element discretization of the auxiliary PDE problems yields a system of algebraic equations that is equivalent to the linear system arising from the FEM discretization of the original PDE problem. The auxiliary linear systems actually form a particular row–wise partition of the original discretization matrix.

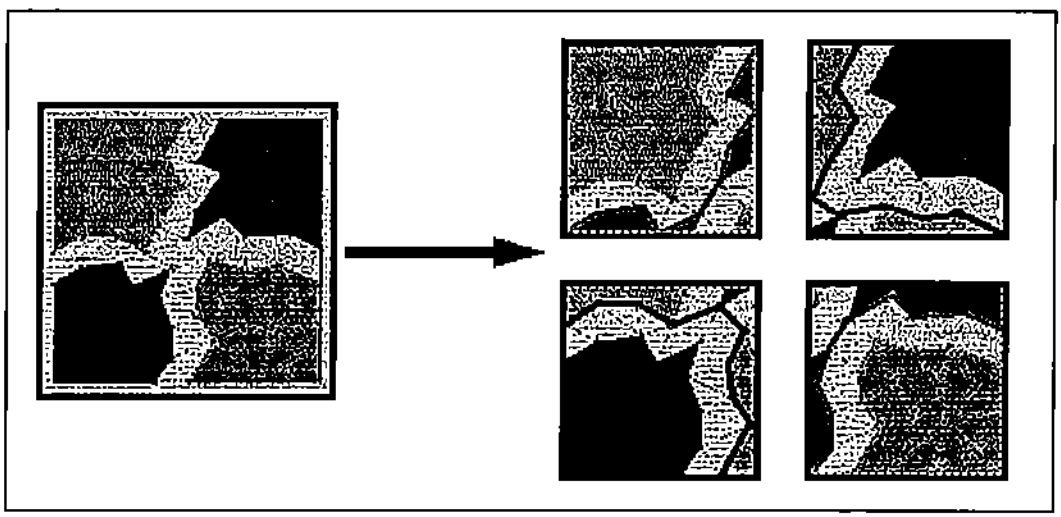

Figure 3: *An example of a discrete domain decomposition and the discrete domains of the corresponding auxiliary PDE problems*

6 The Parallel Linear Algebraic Solvers for PDE Equations

All three parallel methodologies depicted in Figure 1 assume the existence of efficient parallel linear sparse solvers implemented on a set of distributed algebraic data-structures.

There are many parallel solvers that have been proposed and studied in the literature. One class of solvers consists of the classical stationary iterative methods. //ITPACK [18] is such a software system for solving sparse systems arising from finite element and finite difference approximations.

6.1 The Parallel ITPACK Library

The parallel ITPACK system consists of seven modules implementing SOR, Jacobi-CG, Jacobi-SI, RSCG, RSSI, SSOR-CG and SSOR-SI under different indexing schemes [20]. It is integrated in the parallel ELLPACK PSE and is applicable to any linear system stored in parallel ELLPACK's distributed storage scheme. The interfaces of the parallel modules and the assumed data structures are presented in [19]. The library has been proven to be very efficient for elliptic PDEs [18].

The code is based on the sequential version of ITPACK which was parallelized by utilizing a subset of level two sparse BLAS routines [19]. Thus the theoretical behavior of the solver modules remain unchanged from the sequential version. The parallelization is based on the message passing paradigm. The implementation assumes a row–wise splitting of the algebraic equations (obtained indirectly from a non–overlapping decomposition of the PDE domain). Each parallel processor stores a row block of coupled and uncoupled algebraic equations and the requisite communication information, in its local memory.

The communication module of the parallel ITPACK library has been implemented on several MIMD platforms using both native and portable communication libraries. The implementations utilize non–blocking send/receive, reduction and broadcast communication operations from the message passing communication libraries. Parallel ITPACK implementations are available on the Intel Paragon, Intel iPSC 860 and nCUBE 2 parallel machines, as well as on workstation clusters. It has been implemented for these MIMD platforms using MPI [1], [2], [9], [10], PVM [7], PICL [8] and PARMACS [12] portable communication libraries as well as native communication libraries of the parallel machines [4], [25].

7 Computational Performance Evaluation

In this section we attempt to estimate the overhead of the parallel reuse frameworks depicted in Figure 1. We also evaluate the performance of the parallel ITPACK solvers on several MIMD platforms using different portable communication libraries.

7.1 Performance of the Parallel Reuse Methodologies

We used the Parallel ELLPACK PSE to specify the following PDE model and measure the execution times of the PDE solvers.

The temperature distribution on a two dimensional slice of a reactor with a steel dome and concrete base (Figure 4) is computed. The inside surface of the dome is initially 450°K and the ambient temperature around the reactor is 80°K. It is assumed that no heat is lost through the bottom surface of the dome or base. Since the problem is symmetric, we consider only the right half of the slice. We want to find the temperature T such that

$$\nabla \cdot k \nabla T = 0, x \in \Omega.$$

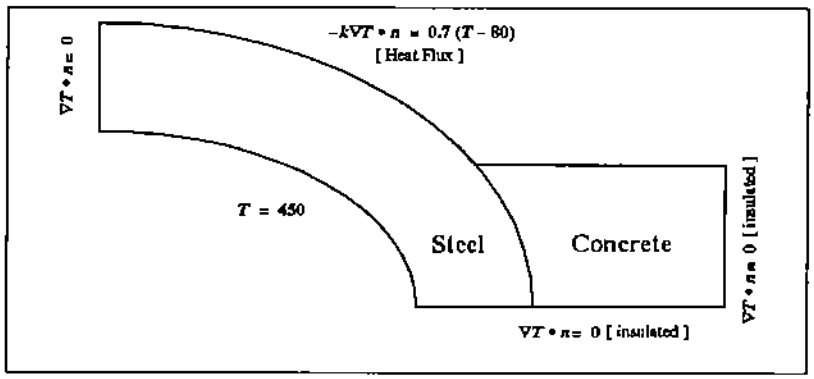

Figure 4: *Two-dimensional domain of the PDE model for steady–state heat diffusion in a reactor*

Here k is the thermal conductivity, $k = 30.62$ in Ω_1 (steel) and $k = 0.79$ in Ω_2 (concrete). The boundary conditions for the interior and exterior surfaces are

$$\nabla T \cdot \mathbf{n} = 0 T = 450 - k\nabla T \cdot \mathbf{n} = 0.7(T - 80)$$

where $\mathbf{n}$ is the exterior unit outward normal to $\partial\Omega$.

The PDE problem domain shown in Figure 4 was discretized with a triangular finite element mesh of 4477 nodes and 8612 elements. The system of the discrete equations was solved by the Jacobi–CG parallel ITPACK solver [18]. Table 3 compares the execution time for the custom parallel FEM discretization and the D^+ discretization methodology.

Table 4 shows the execution time ratios of the D^+ and M^+ discretization methodologies with respect to the custom parallel FEM discretization. They illustrate the low overhead of the D^+ reuse methodology in comparison with the customized parallel discretization methodology. Furthermore the relative ease of implementation of the D^+ methodology, convincingly indicates its effectiveness as a good framework for the parallel reuse of "legacy" elliptic PDE solver software. Figure 5 illustrates the effectiveness of the M^+ methodology in comparison with the sequential method.

Reuse Approach	Machine Configuration					
	1	2	4	8	16	32
Custom Parallel	6.12	3.32	1.87	0.91	0.58	0.30
D^+	6.26	3.47	1.96	1.04	0.61	0.32

Table 3: *Execution time on the Intel Paragon for the custom parallel FEM discretization and the D^+ discretization methodology*

Reuse Approach	Machine Configuration					
	1	2	4	8	16	32
D+ (on–line)	1.03	1.03	1.03	1.04	1.05	1.06
M+ (off–line)	1.03	1.17	3.85	7.41	13.82	26.05

Table 4: *The disretization time ratios of the "on–line" and "off–line" reuse methodologies with respect to custom parallel methodology on the nCUBE 2*

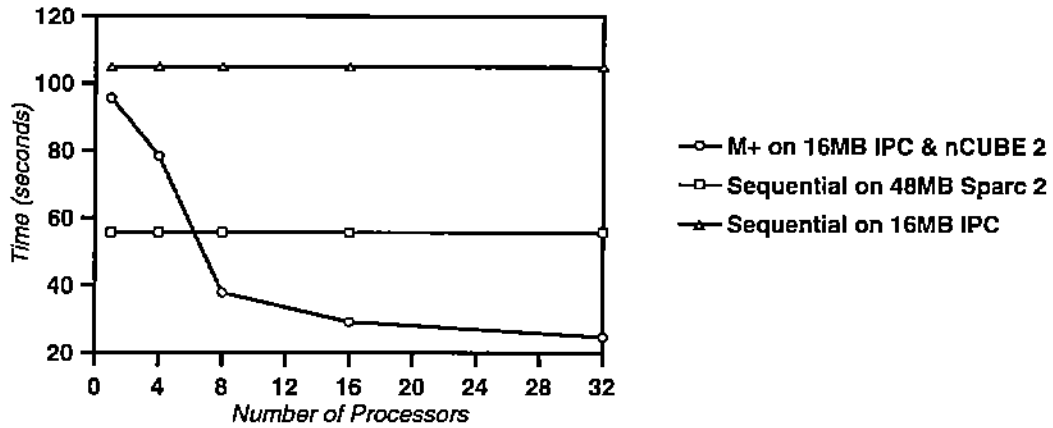

Figure 5: *Total exeution time comparison between the sequential discretization method and the M^+ methodology*

7.2 Performance of the Parallel ITPACK Library

Performance of the parallel ITPACK solvers have been measured on a 64–node nCUBE 2, 16–node iPSC/860, 140–node Paragon and clusters of Sun workstations.

We consider a Helmholtz–type problem

$$u_{xx} + u_{yy} - [100 + \cos(2\pi x) + \sin(3\pi y)]u = f(x, y)$$

with true solution

$$u(x,y) = -0.31[5.4-\cos(4\pi x)]\sin(\pi x)(y^2-y)[5.4-\cos(4\pi y)][(1+(4(x-0.5)^2+4(y-0.5)^2)^4)^{-1}-0.5]$$

and Dirichlet boundary conditions (figure 6).

Parallel ITPACK solvers were used to solve the finite difference equations of this PDE problem. Figure 7 contains the performance measurements of the parallel ITPACK solvers on the Paragon. Figure 8 depicts the comparison of the parallel ITPACK Jacobi CG solver on several hardware platforms and Figure 9 illustrates its speedup for different communication libraries on the nCUBE 2.

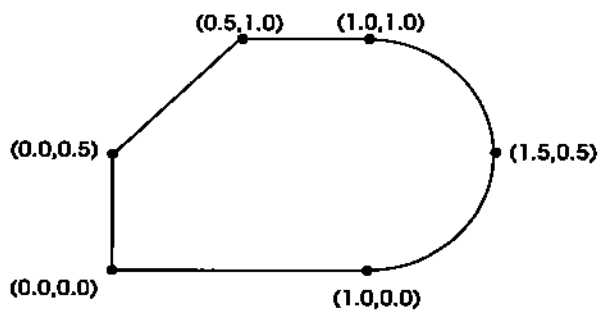

Figure 6: *Domain for the Helmholtz–type boundary value problem with a boundary consisting of lines connecting the points* $(1,0), (0,0), (0,0.5), (0.5,1)$ *and* $(1,1)$ *and the half circle* $x = 1+0.5\sin(t), y = 0.5 - 0.5\cos(t), t \in [0,\pi]$

Number of Processors		Jacobi SI		Jacobi CG		SOR	
		150x150	200x200	150x150	200x200	150x150	200x200
1	time	198.61	356.40	80.24	141.41	31.46	73.30
	speedup	1.00	1.00	1.00	1.00	1.00	1.00
2	time	103.80	183.46	42.07	73.73	16.33	37.93
	speedup	1.91	1.94	1.91	1.92	1.93	1.94
4	time	61.07	107.03	25.09	43.52	9.51	22.22
	speedup	3.25	3.33	3.20	3.25	3.31	3.30
8	time	32.83	56.46	14.29	23.75	5.03	11.68
	speedup	6.05	6.31	5.61	5.95	6.26	6.29
16	time	18.38	30.51	8.85	13.59	2.94	6.36
	speedup	10.80	11.68	9.06	10.41	10.70	11.55
32	time	11.26	17.22	6.38	8.66	1.85	3.67
	speedup	17.63	20.70	12.57	16.33	16.97	20.00
64	time	8.16	11.33	7.59	6.60	1.43	2.51
	speedup	24.35	31.47	10.57	21.43	21.97	29.26

Figure 7: *Performance of Parallel ITPACK iterative solvers on the Paragon*

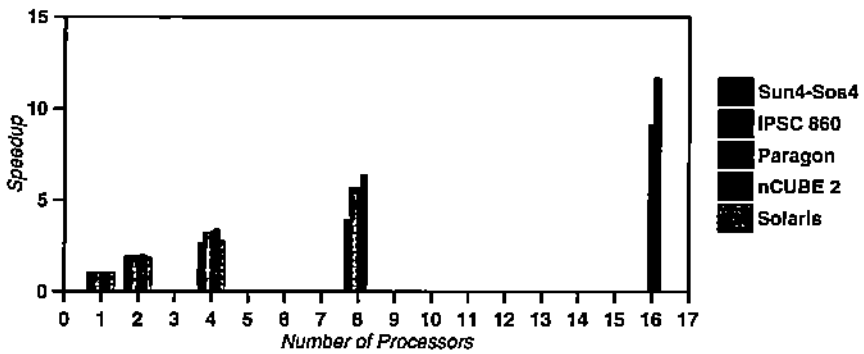

Figure 8: *Speedup Comparison of MPICH–MPI based parallel ITPACK Jacobi CG solver on different hardware platforms*

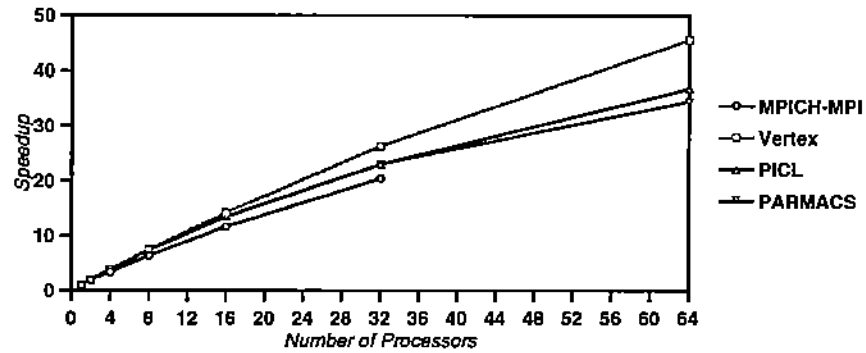

Figure 9: *Speedup Comparison of different communication library implementations of the parallel ITPACK Jacobi CG solver on the nCUBE 2*

8 Conclusion

The parallel reuse methodologies described in this article were implemented and tested for "legacy" finite element code. Experimental results indicate the effectiveness of these parallel reuse frameworks. We plan to extend the scope of these methodologies to include finite difference discretization schemes.

The parallel ITPACK experimental performance data on multiple MIMD platforms indicate its efficiency. These results provide a basis for the selection of the best communication library and hardware platform combination, for the parallel solution of elliptic PDE discretization systems. We also plan on using the parallel ITPACK experimental performance data to compare and evaluate different portable message passing library implementations.

References

[1] R. A. Bruce, J. G. Mills, and A. G. Smith. Chimp/mpi user guide. Technical Report EPCC-KTP-CHIMP-V2-USER 1.2, University of Edinburgh, UK.

[2] G. Burns, R. Daoud, and J. Vaigl. Lam: An open cluster environment for mpi. *Ohio Supercomputing Center, Ohio.*

[3] N. P. Chrisochoides, M. Aboelaze, E. N. Houstis, and C. E. Houstis. The parallelization of some level 2 and 3 blas operations on distributed-memory machines. In *Advances in Computer Methods for Partial Differential Equations*, pages 127–133. IMACS, 1992.

[4] Intel Corporation. iPSC/860: System user's guide. March 1992.

[5] C. Farhat and H. D.Simon. TOP/DOMDEC - a software tool for mesh partitioning and parallel processing. Technical Report RNR-93-011, pp.1-28, NASA Ames Research Center, 1993.

[6] C. Farhat and E. Wilson. A new finite element concurrent computer program architecture. *Int. J. for Numerical Methods in engineering*, 24(0):1771–1792, 1987.

[7] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidyalingam S. Sunderam. *PVM: A Users' Guide and Tutorial for Network Parallel Computing.* MIT Press, 1994.

[8] G. A. Geist, M. T. Heath, B.W. Peyton, and P.H. Worley. A user's guide to picl: A portable instrumented communication library. Technical Report ORNL/TM-11616, Engineering Physics and Mathematics Division, Oak Ridge National Laboratory, 1992.

[9] B. Gropp, R. Lusk, T. Skjellum, and N. Doss. Portable MPI Model Implementation. Technical Report No., Argonne National Laboratory, July 1994.

[10] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface.* MIT Press, October 1994.

[11] L. Gross, C. Roll, and W. Schoenauer. Vecfem for mixed finite elements. Technical Report Interner Bericht Nr. 50/93, Rechenzentrum der Universitat Karlsruhe, December 1993.

[12] R. Hempel, H.-C. Hoppe, and A. Supalov. A proposal for a parmacs library interface. Technical Report GMD, Postfach 1316, D-5205 Sankt Augustin 1, Germany, October 1992.

[13] E. N. Houstis, S. B. Kim, S. Markus, P. Wu, N. E. Houstis, A. C. Catlin, S. Weerawarana, and T. S. Papatheodorou. Parallel ELLPACK PDE Solvers. *Intel SuperComputer User's Group Conference*, 1995.

[14] E. N. Houstis, T. S. Papatheodorou, and J. R. Rice. Parallel ELLPACK: An expert system for the parallel processing of partial differential equations. In *Intelligent Mathematical Software Systems*, pages 63–73. North-Holland, Amsterdam, 1990.

[15] E. N. Houstis and J. R. Rice. Parallel ellpack: A development and problem solving environment for high performance computing machines. In P. W. Gaffney and E. N. Houstis, editors, *Programming Environments for High-Level Scientific Problem Solving*, pages 229–241. North-Holland, 1992.

[16] E. N. Houstis, J. R. Rice, N. P. Chrisochoides, H. C. Karathanasis, P. N. Papachiou, M. K. Samartzis, E. A. Vavalis, K. Y. Wang, and S. Weerawarana. Ellpack: A numerical simulation programming environment for parallel mimd machines. In D. Marinescu and R. Frost, editors, *International Conference on Supercomputing*, pages 96–107, Amsderdam, June 1990. ACM Press NY.

[17] Claes Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, 1987.

[18] S. Kim, E. N. Houstis, and J. R. Rice. Parallel stationary iterative methods and their performance. In D. Marinescu and R. Frost, editors, *INTEL supercomputer users group conference*, San Diego, 1994.

[19] Sang-Bae Kim. Parallel Numerical Methods for Partial Differential Equations. Ph.D. Thesis, 1993, Department of Mathematics, Purdue University. Technical Report CSD-TR-94-000, pp.1-00, Purdue University, Computer Science, 1993.

[20] D. Kinkaid, J. Respess, and R. Grimes. Algorithm 586: Itpack 2c: A fortran package for solving large linear systems by adaptive accelerated iterative methods. *ACM Tran. Math. Soft.*, 8(0):302–322, 1982.

[21] N.K. Madsen (Lawrence Livermore Laboratories) and R.F. Sincovec (Kansas State University). Algorithm 540: Pdecol, general sollocation software for partial differential equations. *ACM Transactions on Mathematical Software*, 5(3):326–351, September 1979.

[22] J.G. Malone. Automated mesh decomposition and concurrent finite element analysis for hypercube multiprocessor computers. *Computer methods in applied mechanics and engineering*, 70(0):27–58, 1988.

[23] S. Markus, A. C. Catlin, S. Weerawarana, and E. N. Houstis. On the software engineering of multi-platform parallel/distributed software. *Proc. of First International Conference in Neural, Parallel and Scientific Computation*, 1995.

[24] W. F. Mitchell. Adaptive refinement for arbitrary finite element spaces with hierarchical bases. *J. Computational and Applied Math.*, 36:65–78, 1991.

[25] nCUBE Corporation. nCUBE 2 programmer's guide. *Release 2.0*, 1990.

[26] B. Nour-Omid, A. Raefsky, and G. Lyzenga. Solving finite element equations on concurrent computers. In A. Noor, editor, *Parallel computations and their impact on mechanics*, pages 209–226, 1987.

[27] J. R. Rice and R. F. Boisvert. *Solving elliptic problems using ELLPACK.* Springer-Verlag, 1985.

[28] M. Schmauder, R. Weiss, and W. Schoenauer. The cadsol program package. Technical Report Interner Bericht Nr. 46/92, Rechenzentrum der Universitat Karlsruhe, April 1992.

[29] W. Schoenauer, E. Schnepf, and H. Mueller. The fidisol program package. Technical Report Interner Bericht Nr. 27/85, Rechenzentrum der Universitat Karlsruhe, December 1985.

[30] P. Le Tallec, Y. H. De Roeck, and M. Vidrascu. Domain decomposition methods for large linearly elliptic three-dimensional problems. *J. Computational and Applied Math.*, 34:93–117, 1991.

[31] David K. Melgaard (Kansas State University) and Richard F. Sincovec (Boeing Computer Services Company). General software for two-dimensional nonlinear partial differential equations. *ACM Transactions on Mathematical Software*, 7(1):106–125, March 1981.

[32] Richard F. Sincovec (Kansas State University) and Niel K. Madsen (Lawrence Livermore Laboratories). Software for nonlinear partial differential equations. *ACM Transactions on Mathematical Software*, 1(3):232–260, September 1975.

[33] V. Venkatakrishman, H. D. Simon, and T. J. Barth. A mimd implementation of a parallel euler solver for unstructured grids. *The Journal of Supercomputing*, 6(0):117–137.

[34] G. Yagawa, N. Soneda, and S. Yoshimura. A large scale finite element alalysis using domain decomposition method on a prallel computer. *Computers and structures*, 38(5/6):615–625, 1991.