International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:1, No:4, 2007

Grid Computing for the Bi-CGSTAB applied to the solution of the Modified Helmholtz Equation

E. N. Mathioudakis and E. P. Papadopoulou

Abstract**—** The problem addressed herein is the efficient management of the Grid/Cluster intense computation involved, when the preconditioned Bi-CGSTAB Krylov method is employed for the iterative solution of the large and sparse linear system arising from the discretization of the Modified Helmholtz-Dirichlet problem by the Hermite Collocation method. Taking advantage of the Collocation matrix's red-black ordered structure we organize efficiently the whole computation and map it on a pipeline architecture with master-slave communication. Implementation, through MPI programming tools, is realized on a SUN V240 cluster, interconnected through a 100Mbps and 1Gbps ethernet network, and its performance is presented by speedup measurements included.

Keywords**—** Collocation, Preconditioned Bi-CGSTAB, MPI, Grid and DSM Systems.

I. Introduction

Hermite Collocation is a high order finite element scheme used as a discretizer especially when continuous first derivatives are required. Among other properties, Collocation produces large and sparse systems of equations which poses no pleasant properties (e.g. symmetry). Memory requirements and performance are two of the main factors suggesting the usage of iterative methods on multiprocessor environments [2]. This motivated relevant research in the areas of iterative method analysis and parallel algorithm development. Main issues addressed were concerning both algorithmic (multi-color orderings, domain decomposition/partitioning techniques, parallel preconditioners, etc) and architectural (memory management/distribution, processor architecture, etc) aspects.

Manuscript received May 10, 2007

E. N. Mathioudakis is with the Department of Sciences, Technical University of Crete, Campus, 73100 Chania, Greece (phone: +30-28210-37750; fax: +30-28210-37842; e-mail: manolis@science.tuc.gr).

E. P. Papadopoulou is with the Department of Sciences, Technical University of Crete, Campus, 73100 Chania, Greece (e-mail: elena@science.tuc.gr).

Particular results, in this direction, concerning the Collocation method may be found in (e.g. [1],[3]-[5],[14] and in our work in [6]-[11],[13]. In [11] we considered the implementation of the SOR and the Bi-CGSTAB iterative methods for solving the Collocation system for elliptic problems on Distributed-Shared memory (DSM) machines, improving the overall performance by managing the whole computation in order to maximize locality and minimize communication among the processing elements.

The work herein extends the results in [11] by considering :

- the Modified Helmoltz operator as our model problem
- a Grid/Clustered computational environment, thus introducing the additional parameter of interconnecting the processors through a common local ethernet-type network.

This paper is organized as follows: In Sections 2 and 3 we briefly describe the structure of the collocation system, the preconditioned Bi-CGSTAB [17] algorithm, and the basic features of the parallel architecture used to carry out the whole computation. In Section 4, we present the performance measurements from the implementation on a SUN V240 [16] clustered system and, for comparison purposes, on a SGI Origin 350 [15] DSM machine.

II. Collocation for the Helmholtz BVP

To fix notation, consider the Modified Helmholtz Dirichlet Boundary Value Problem (BVP):

$$\begin{cases} \nabla^2 u(x,y) - \lambda u(x,y) &= f(x,y)\ ,\ (x,y) \in \Omega \\ u(x,y) &= g(x,y)\ ,\ (x,y) \in \partial\Omega \end{cases} \quad (1)$$

with $\lambda \geq 0$, on the rectangular domain $\Omega \equiv (0,1) \times (0,1)$. Assuming a uniform partition of the interval $[0,1]$ into $n_s = 2p$ subintervals, the Hermite Bi-Cubic approximation seeks an approximate solution $\tilde{u}(x,y)$

in the form

$$u(x,y) \sim \tilde{u}(x,y) = \sum_{i=1}^{\tilde{n}}\sum_{j=1}^{\tilde{n}} \alpha_{i,j}\phi_i(x)\phi_j(y)\,, \quad (2)$$

where $\tilde{n} = 2(n_s+1)$ and the basis functions $\phi_i(x)$ and $\phi_j(y)$ are the well known one dimensional piecewise Hermite cubic polynomials. The collocation equations needed for the determination of the $n = 4n_s^2$ unknowns are constructed by forcing the approximate solution $\tilde{u}(x,y)$ to satisfy the BVP in n interior collocation points. These are the four Gauss points in each of the n_s^2 elements, a classical choice for orthogonal spline Collocation. In doing so one obtains the $n \times n$ linear system of equations

$$A\boldsymbol{x} = \boldsymbol{b}\,, \quad (3)$$

where A is the *Collocation coefficient matrix* and

$$\boldsymbol{x} = [x_1 \;\; x_2 \;\; \cdots \;\; x_n]^T \equiv [\alpha_{1,1} \;\; \cdots \;\; \alpha_{\tilde{n},\tilde{n}}]^T$$

is the unknown vector. Collocation method possesses no restrictions on how one orders or numbers the equations and the unknowns for the construction of the Collocation system. However, the *block structure* of the collocation matrix depends directly on this numbering. A particular *Red-Black* numbering, used also in [10] (see also [9] for a complete description), leads to the block structure of the Collocation matrix

$$A = \begin{pmatrix} D_R & H_B \\ H_R & D_B \end{pmatrix}\,, \quad (4)$$

where, the matrices D_R, H_B, H_R and D_B have exactly the same structure with the corresponding matrices in [10],[11] with some certain deviations due to the Helmholtz operator. That is to say,

$$D_R = \text{diag}[\underbrace{A_2 \;\; 2A_1 \;\; 2A_2 \;\; \cdots \;\; 2A_1 \;\; 2A_2 \;\; -A_2}_{2p-blocks}]\,, \quad (5)$$

$$D_B = 2\,\text{diag}[\underbrace{A_1 \;\; A_2 \;\; \cdots \;\; A_1 \;\; A_2}_{2p-blocks}] \quad (6)$$

$$H_R = \begin{pmatrix} R_1 & R_2 & & & & \\ R_3 & R_1 & R_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & R_3 & R_1 & R_2 \\ & & & & R_3 & \hat{R}_1 \end{pmatrix} \quad (7)$$

$$H_B = \begin{pmatrix} B_1 & B_2 & & & & \\ B_3 & B_1 & B_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & B_3 & B_1 & B_2 \\ & & & & B_3 & B_1 \end{pmatrix} \quad (8)$$

where

$$R_1 = \begin{pmatrix} A_4 & A_3 \\ -A_4 & A_3 \end{pmatrix}\,,\; \hat{R}_1 = \begin{pmatrix} A_4 & -A_4 \\ -A_4 & -A_4 \end{pmatrix}\,,$$

$$R_2 = -\begin{pmatrix} A_4 & 0 \\ A_4 & 0 \end{pmatrix}\,,\; R_3 = \begin{pmatrix} 0 & A_3 \\ 0 & -A_3 \end{pmatrix}\,,$$

and

$$B_1 = \begin{pmatrix} A_3 & -A_4 \\ A_3 & A_4 \end{pmatrix}\,,$$

$$B_2 = \begin{pmatrix} 0 & 0 \\ A_3 & -A_4 \end{pmatrix}\,,\; B_3 = -\begin{pmatrix} A_3 & A_4 \\ 0 & 0 \end{pmatrix}\,.$$

The $2n_s \times 2n_s$ matrices A_1, A_2, A_3 and A_4 are banded (with bandwidth 5) and their structure is given by

$$\begin{pmatrix} a_2 & a_3 & -a_4 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ a_4 & a_1 & -a_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_3 & -a_4 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & a_3 & a_4 & a_1 & -a_2 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_1 & a_2 & a_3 & -a_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_3 & a_4 & a_1 & -a_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_1 & a_2 & -a_4 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_3 & a_4 & -a_2 \end{pmatrix}$$

where the values of a_i's are defined by

	a_1	a_2	a_3	a_4
A_1	r^+	s^+	q	t^+
A_2	s^+	u^+	t^-	ϵ
A_3	q	t^-	r^-	s^-
A_4	t^+	ϵ	s^-	u^-

with

$\epsilon = -\frac{\lambda}{24n_s^2}$, $q = 24 + 22\epsilon$,
$r^{\pm} = 86\epsilon - 24 \pm (48\epsilon - 18)\sqrt{3}$,
$s^{\pm} = 13\epsilon - 12 \pm (7\epsilon - 8)\sqrt{3}$,
$t^{\pm} = 5\epsilon + 3 \pm (\epsilon + 1)\sqrt{3}$,
$u^{\pm} = 2\epsilon - 3 \pm (\epsilon - 2)\sqrt{3}$.

III. Parallel Bi-CGSTAB for Collocation

Consider the classical splitting of the matrix A in (4) as

$$A = D_A - L_A - U_A \tag{9}$$

where
$$D_A = \begin{bmatrix} D_R & O \\ O & D_B \end{bmatrix} \ , \ L_A = \begin{bmatrix} O & O \\ -H_R & O \end{bmatrix} \tag{10}$$

and

$$U_A = \begin{bmatrix} O & -H_B \\ O & O \end{bmatrix} . \tag{11}$$

Then, the algorithm for the B-GS preconditioned Bi-CGSTAB [17] method is described by :

Choose $\boldsymbol{x}^{(0)}$
$\boldsymbol{r^{(0)}} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$
Choose $\hat{\boldsymbol{r}}$ (usually $\hat{\boldsymbol{r}} = \boldsymbol{r^{(0)}}$)
for $i = 1, 2, ...$
  $\rho_{i-1} = \hat{\boldsymbol{r}}^T \boldsymbol{r}^{(i-1)}$
  if $\rho_{i-1} = 0$ method **fails**
  if $i = 1$
    $\boldsymbol{p^{(1)}} = \boldsymbol{r^{(0)}}$
  else
    $\beta_{i-1} = \frac{\rho_{i-1}}{\rho_{i-2}} \ \frac{\alpha_{i-1}}{\omega_{i-1}}$
    $\boldsymbol{p}^{(i)} = \boldsymbol{r}^{(i-1)} + \beta_{i-1}(\boldsymbol{p}^{(i-1)} - \omega_{i-1}\, \boldsymbol{v}^{(i-1)})$
  endif
  Solve $\mathcal{M}\, \hat{\boldsymbol{p}} = \boldsymbol{p}^{(i)}$; $\boldsymbol{v}^{(i)} = A\, \hat{\boldsymbol{p}}$
  $\alpha_i = \frac{\rho_{i-1}}{\hat{\boldsymbol{r}}^T \boldsymbol{v}^{(i)}}$
  $\boldsymbol{s} = \boldsymbol{r}^{(i-1)} - \alpha_i\, \boldsymbol{v}^{(i)}$
  if $\| \boldsymbol{s} \|$ is small enough **then**
    $\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i\, \hat{\boldsymbol{p}}$ and **stop**
  Solve $\mathcal{M}\, \boldsymbol{z} = \boldsymbol{s}$; $\boldsymbol{t} = A\, \boldsymbol{z}$
  $\omega_i = \frac{\boldsymbol{s}^T \boldsymbol{t}}{\boldsymbol{t}^T \boldsymbol{t}}$
  $\boldsymbol{x}^{(i)} = \boldsymbol{x}^{(i-1)} + \alpha_i\, \hat{\boldsymbol{p}} + \omega_i\, \boldsymbol{z}$
  Check for Convergence
  if $\omega_i = 0$ **stop**
  $\boldsymbol{r}^{(i)} = \boldsymbol{s} - \omega_i\, \boldsymbol{t}$
end

As in [10],[11], the preconditioner matrix $\mathcal{M}$ is the splitting matrix of the Backward Gauss-Seidel method (B-GS) defined by

$$\mathcal{M} = D_A - U_A \ . \tag{12}$$

Naturally, one now has to incorporate the particular structures, of the matrices involved, into the algorithm (see [11]. For example, in doing so for the computationally intense statement

Solve $\mathcal{M}\, \boldsymbol{z} = \boldsymbol{s}$; $\boldsymbol{t} = A\, \boldsymbol{z}$

we obtain:

Solve $D_B\, \boldsymbol{z}_B = \boldsymbol{s}_B$
$\boldsymbol{y} = H_B\, \boldsymbol{z}_B$
Solve $D_R\, \boldsymbol{z}_R = \boldsymbol{s}_R - \boldsymbol{y}$
$\hat{\boldsymbol{y}} = H_R\, \boldsymbol{z}_R$
$\boldsymbol{t}_R = \boldsymbol{s}_R$
$\boldsymbol{t}_B = \boldsymbol{s}_B + \hat{\boldsymbol{y}}$

Following the analysis in [11], our work in [8],[9], and taking into consideration the essential factors of (a) uniform load balancing, (b) minimal idle cycles of processors, and (c) minimal communication cost, we consider a *pipelined* architecture consisting of $\mathcal{P}_j$, $j = 1, \cdots, N$ processors. Each processor is assigned to execute the same instruction set which, for the above mentioned computationally intense statement takes the form [11]:

Black Cycle

do $l = 2p + (j-1)k + 1$ to $2p + jk - 1$, 2
  Solve $2A_2 \boldsymbol{z}_l^{(B)} = \boldsymbol{s}_l^{(B)}$
  $\boldsymbol{y}_{l-2p} = A_3 \boldsymbol{z}_l^{(B)}$
  Solve $2A_1 \boldsymbol{z}_{l+1}^{(B)} = \boldsymbol{s}_{l+1}^{(B)}$
  $\boldsymbol{y}_{l+1-2p} = A_4 \boldsymbol{z}_{l+1}^{(B)}$
enddo
$\begin{bmatrix} \boldsymbol{tc}_1 \\ \boldsymbol{tc}_2 \end{bmatrix} \leftarrow$Receive $\begin{bmatrix} \boldsymbol{y}_{(j-1)k-1} \\ \boldsymbol{y}_{(j-1)k} \end{bmatrix}$ from $\mathcal{P}_{j-1}$
Send to $\mathcal{P}_{j+1}$ $\begin{bmatrix} \boldsymbol{y}_{jk-1} \\ \boldsymbol{y}_{jk} \end{bmatrix}$
Send to $\mathcal{P}_{j-1}$ $\begin{bmatrix} \boldsymbol{y}_{(j-1)k+1} \\ \boldsymbol{y}_{(j-1)k+2} \end{bmatrix}$
$\begin{bmatrix} \boldsymbol{tc}_3 \\ \boldsymbol{tc}_4 \end{bmatrix} \leftarrow$Receive $\begin{bmatrix} \boldsymbol{y}_{jk+1} \\ \boldsymbol{y}_{jk+2} \end{bmatrix}$ from $\mathcal{P}_{j+1}$
$\boldsymbol{tm}_1 \leftarrow \boldsymbol{tc}_1 + \boldsymbol{tc}_2$
$\boldsymbol{tm}_2 \leftarrow \boldsymbol{y}_{(j-1)k+1} - \boldsymbol{y}_{(j-1)k+2}$
do $l = (j-1)k + 1$ to $jk - 3$, 2
  $\boldsymbol{tm}_3 \leftarrow \boldsymbol{y}_l$
  $\boldsymbol{y}_l \leftarrow \boldsymbol{tm}_2 - \boldsymbol{tm}_1$
  $\boldsymbol{tm}_1 \leftarrow \boldsymbol{y}_{l+1} + \boldsymbol{tm}_3$
  $\boldsymbol{tm}_2 \leftarrow \boldsymbol{y}_{l+2} - \boldsymbol{y}_{l+3}$
  $\boldsymbol{y}_{l+1} \leftarrow \boldsymbol{tm}_1 + \boldsymbol{tm}_2$
enddo
$\boldsymbol{tm}_3 \leftarrow \boldsymbol{y}_{jk-1} + \boldsymbol{y}_{jk}$
$\boldsymbol{y}_{jk-1} \leftarrow \boldsymbol{tm}_2 - \boldsymbol{tm}_1$
$\boldsymbol{y}_{jk} \leftarrow \boldsymbol{tm}_3 + \boldsymbol{tc}_3 - \boldsymbol{tc}_4$

Red Cycle

do $l = (j-1)k+1$ to $jk-1$, 2
  Solve $2A_1 \boldsymbol{z}_l^{(R)} = \boldsymbol{s}_l^{(R)} - \boldsymbol{y}_l$
  $\boldsymbol{y}_l = A_3 \boldsymbol{z}_l^{(R)}$
  Solve $2A_2 \boldsymbol{z}_{l+1}^{(R)} = \boldsymbol{s}_{l+1}^{(R)} - \boldsymbol{y}_{l+1}$
  $\boldsymbol{y}_{l+1} = A_4 \boldsymbol{z}_{l+1}^{(R)}$
enddo
$\begin{bmatrix} \boldsymbol{tc}_1 \end{bmatrix} \leftarrow$Receive $\begin{bmatrix} \boldsymbol{y}_{(j-1)k} \end{bmatrix}$ from $\mathcal{P}_{j-1}$
Send to $\mathcal{P}_{j+1}$ $\begin{bmatrix} \boldsymbol{y}_{jk} \end{bmatrix}$
Send to $\mathcal{P}_{j-1}$ $\begin{bmatrix} \boldsymbol{y}_{(j-1)k+1} \end{bmatrix}$
$\begin{bmatrix} \boldsymbol{tc}_2 \end{bmatrix} \leftarrow$ Receive $\begin{bmatrix} \boldsymbol{y}_{jk+1} \end{bmatrix}$ from $\mathcal{P}_{j+1}$
$\boldsymbol{tm}_1 \leftarrow \boldsymbol{tc}_1$
do $l = (j-1)k+1$ to $jk-3$, 2
  $\boldsymbol{tm}_2 \leftarrow \boldsymbol{y}_l + \boldsymbol{tm}_1$
  $\boldsymbol{tm}_3 \leftarrow \boldsymbol{y}_{l+1} - \boldsymbol{y}_{l+2}$
  $\boldsymbol{y}_l \leftarrow \boldsymbol{tm}_2 + \boldsymbol{tm}_3$
  $\boldsymbol{tm}_1 \leftarrow \boldsymbol{y}_{l+1}$
  $\boldsymbol{y}_{l+1} \leftarrow \boldsymbol{tm}_3 - \boldsymbol{tm}_2$
enddo
$\boldsymbol{tm}_2 \leftarrow \boldsymbol{y}_{jk-1} + \boldsymbol{tm}_1$
$\boldsymbol{tm}_1 \leftarrow \boldsymbol{y}_{jk} - \boldsymbol{tc}_2$
$\boldsymbol{y}_{jk-1} \leftarrow \boldsymbol{tm}_1 + \boldsymbol{tm}_2$
$\boldsymbol{y}_{jk} \leftarrow \boldsymbol{tm}_1 - \boldsymbol{tm}_2$

We remark that one of the Processors, say $\mathcal{P}_1$, in addition to the computational tasks assigned to each processor, has been also assigned the tasks of *gathering* partially processed data, assemble, in the sequel, the final values for the inner products and other parameters of the algorithm, and finally *broadcast* the results to all other processors.

IV. Realization on a Grid/Clustered System

In this section we present the performance measurements of the implementation of the above parallel algorithm for the test Dirichlet Helmholtz problem which accepts the following exact solution (Fig. 1)

$$u(x,y) = 10\ \phi(x)\ \phi(y)\ ,\quad \phi(x) = e^{-100(x-0.1)^2}(x^2 - x).$$

This implementation was realized on a four-node SUN V240z [16] cluster interconnected through a 100Mbps and 1Gbps ethernet network. Each node consists of dual 1.5 GHz UltraSPARC IIIi processors with each having

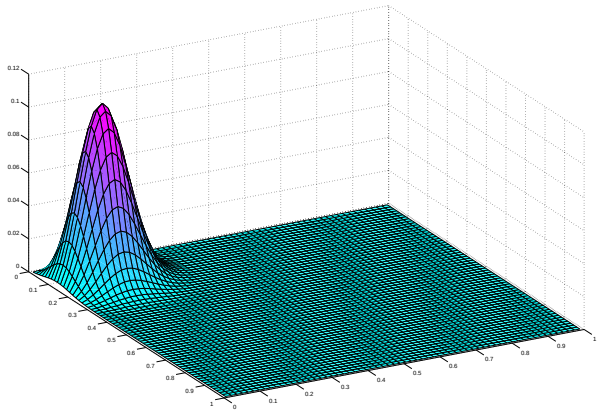

Fig. 1 : The exact solution $u(x,y)$ of the test problem.

1MB L2 cache memory. The total memory of each node is 2GB and the operating system is Solaris 10. The application is developed in double precision Fortran code using the MPI [12] standard for SUN Studio compiler version 10, which also incorporates the scientific library LAPACK. The application is also realized on a Distributed Shared memory machine SGI origin 350 [15] for comparison purposes. SGI Origin 350 is a cache coherent - nonuniform memory access (ccNUMA) architecture machine, consisting of eight R16000@600MHz type processors with 4 MB L2 cache memory each. The total memory is 4 GB and the operating system is IRIX version 6.5 with MipsPro compilers version 7.4.

Tables T1-T3 below summarize the behavior of the method for representative values of λ, namely $\lambda = 0, 1$ and 100, and for several discretization sizes from $n_s = 16$ up to $n_s = 512$ subintervals. At this point we remark that the Bi-CGSTAB converges fast for small to medium values of the Helmholtz parameter λ, while for large values of λ, where the part of the Helmholtz operator involving λ takes over, the convergence rate of the method improves significantly. Moreover, the number of iterations needed for convergence of the method increases in complete analogy to the discretization size characterized by the value of n_s. The theoretical support to these observations is beyond the scope of this work and will be presented elsewhere.

Focusing now on the performance of our parallel algorithm and its implementation on the given grid environment, we note that it is sufficient to describe the results for a typical case of the Helmholtz parameter λ (we have chosen $\lambda = 1$) as they remain independent from its specific values.

T1	Preconditioned Bi-CGSTAB for $\lambda = 0$			
n_s	Iterations	Time	$\| \boldsymbol{u} - \boldsymbol{x} \|_\infty$	$\| \boldsymbol{b} - A\boldsymbol{x} \|_\infty$
16	15	0.009	6.08e-4	3.44e-6
32	27	0.066	3.35e-5	7.55e-7
64	57	0.604	2.02e-6	1.83e-7
128	110	5.751	1.41e-6	8.51e-8
256	216	53.56	2.93e-6	6.45e-8
512	393	415.6	5.07e-6	3.50e-8

T2	Preconditioned Bi-CGSTAB for $\lambda = 1$			
n_s	Iterations	Time	$\| \boldsymbol{u} - \boldsymbol{x} \|_\infty$	$\| \boldsymbol{b} - A\boldsymbol{x} \|_\infty$
16	15	0.009	6.08e-4	2.58e-6
32	27	0.066	3.36e-5	6.92e-7
64	56	0.601	2.23e-6	2.11e-7
128	109	5.811	1.59e-6	9.04e-8
256	206	52.14	2.88e-6	1.39e-7
512	402	431.7	1.39e-6	1.29e-8

T3	Preconditioned Bi-CGSTAB for $\lambda = 100$			
n_s	Iterations	Time	$\| \boldsymbol{u} - \boldsymbol{x} \|_\infty$	$\| \boldsymbol{b} - A\boldsymbol{x} \|_\infty$
16	9	0.006	5.99e-4	4.93e-6
32	16	0.040	3.37e-5	1.19e-6
64	31	0.341	2.01e-6	2.73e-7
128	56	2.987	2.59e-7	8.73e-8
256	108	27.42	4.67e-7	4.39e-8
512	233	249.8	4.47e-7	1.56e-7

The associated to the case of $\lambda = 1$ performance results are summarized in Table T4 and Figures 2-4. Table T4 contains the results pertaining to the computation and communication time independently for $n_s = 64, 256$ and 512, while the case $n_s = 128$ is graphically presented through Figures 2, 3 and 4 for the cases of 100Mbps, 1Gbps network connections and the SGI Origin 350 DSM system respectively. Inspecting Table T4 one may easily observe that the implemented algorithm has efficiently partitioned the whole computation involved such that the computation time appears to decrease nearly exponentially with respect to the number of processors. This is, of course, independent of the network's speed or the DSM system involved in the implementation.

T4		100Mbps		1Gbps	
n_s	Procs	Tcomp	Tcomm	Tcomp	Tcomp
64	2	0.267	0.007	0.267	0.007
	4	0.157	0.247	0.149	0.148
	8	0.074	0.339	0.074	0.219
256	2	30.23	0.156	30.23	0.156
	4	13.09	2.375	13.05	0.825
	8	6.312	2.579	6.306	1.517
512	2	279.0	0.371	279.0	0.371
	4	136.8	2.175	136.7	1.078
	8	58.93	11.34	58.92	6.842

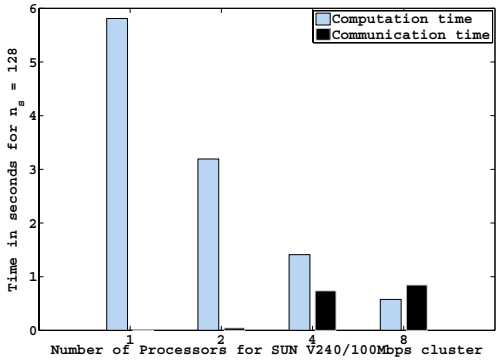

Fig. 2 : Time measurements for SUN V240/100Mbps cluster.

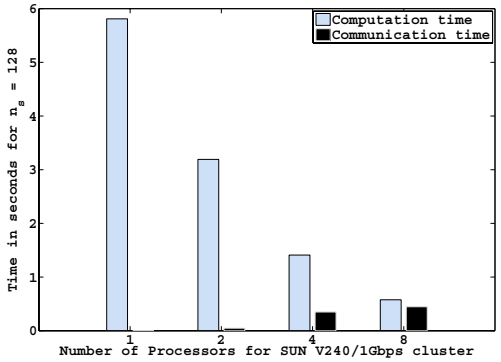

Fig. 3 : Time measurements for SUN V240/1Gbps cluster.

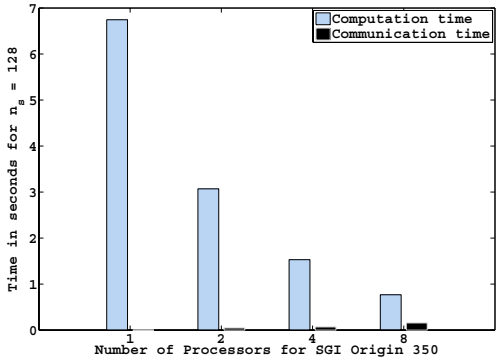

Fig. 4 : Time measurements for SGI Origin 350.

With respect, now, to the communication time we would like at first to make the following general remarks: (a) the time pertaining to the case of two processors remains low in all cases as the processors are on the same grid node, (b) the time increases as the number of processors and the discretization size increase. For slower, though, networks (column 4 of T4 and Fig. 2) and for small to medium discretization sizes (cases of $n_s = 64$ and 128) communication overtakes computation effecting the overall performance of the implementation. This fact is also shown in Figures 5

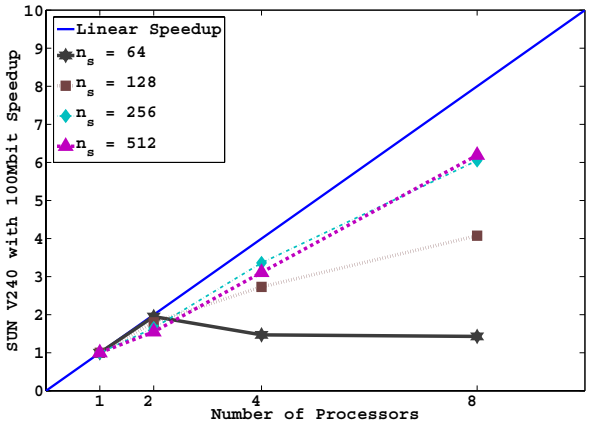

Fig. 5 : Speedup measurements for SUN V240/100Mbps cluster.

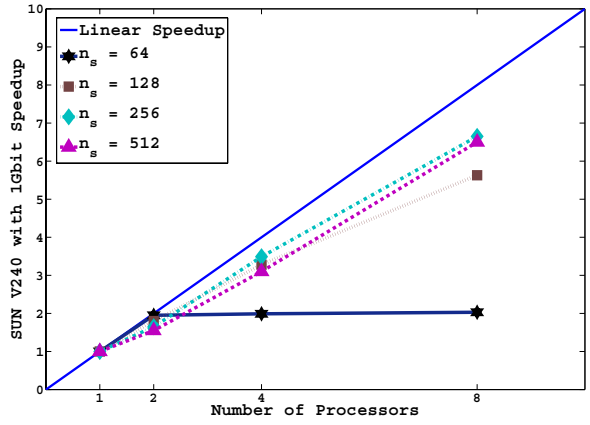

Fig. 6 : Speedup measurements for SUN V240/1Gbps cluster.

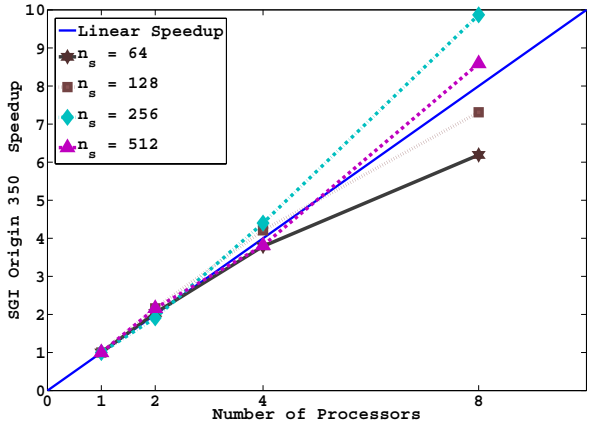

Fig. 7 : Speedup measurements for SGI Origin 350.

and 6, where we present the speedup measurements. It becomes apparent that faster networks yield better performance. The implementation on SGI Origin 350 yields superlinear speedup as the discretization becomes finer, as seen in Figure 7. This is attributed to the fact of the high speed processor interconnection and its associated cache memory.

References

[1] S.H.Brill and G.F.Pinder, "Parallel implementation of the Bi-CGSTAB method with Block Red-Black Gauss-Seidel preconditioner applied to the Hermite Collocation discretization of partial differetial equations",*Parallel Computing*, vol. 28, pp. 399-414, 2002.

[2] J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Toczon and A. White, *SourceBook of Parallel Computing*, San Francisco,CA: Morgan Kaufmann Publishers, 2003.

[3] C.C.Christara, "Parallel solvers for spline collocation equations", *Advances in Eng. Software*,vol. 27, pp.71-89,1996.

[4] J.F. Guarnaccia and G.F. Pinder, "A Collocation based parallel algorithm to solve immiscible two phase flow in porows media", *Procs of the 5th SIAM conf. on Parallel processing for scientific computing*, pp. 205-210, 1992.

[5] C. E. Houstis, E. N. Houstis and J.R. Rice, "Partitioning PDE Computations: Methods and Performance Evaluation", *Parallel Computing*,vol. 5, pp. 141-163, 1997.

[6] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "Mapping Parallel Iterative Algorithms for PDE Computations on a Distributed Memory Computers", *Parallel Alg. and Appl.* , vol. 8, pp. 141-154, 1996.

[7] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "Non-Stationary Iterative Schemes for the Solution of Elliptic Collocation Systems", *Procs of the 4nd Hellenic - European conf. on computer math. and its applic.*, pp. 1044-1051, 1998.

[8] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "Bi-CGSTAB for collocation equations on distributed memory parallel computers" , *Numerical Mathematics and advanced applications - ENUMATH 2001*, pp. 957-966, 2003.

[9] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "Iterative Solution of Elliptic Collocation Systems on a Cognitive Parallel Computer", *Computers and Mathematics with applications*,vol. 48, pp. 951-970, 2004.

[10] E. N. Mathioudakis, E. P. Papadopoulou and Y. G. Saridakis, "Preconditioning for solving Hermite Collocation by the Bi-CGSTAB", *WSEAS Trans. on Mathematics*, vol 5, no. 7, pp. 811-816, July 2006.

[11] E. N. Mathioudakis and E. P. Papadopoulou, "MPI Managment of Hermite Collocation computation on a Distributed-Shared memory system", *WSEAS Trans. on Mathematics*, vol. 5, no. 5, pp. 520-525, May 2006.

[12] Message Passing Interface (MPI) web page, *http://www.mpi-forum.org*

[13] E. P. Papadopoulou, Y. G. Saridakis and T.S. Papatheodorou, "Orderings and Partitions of PDE computations for a fixed size VLSI architecture", *Procs of IEEE-ACM Fall Joint Computer Science Conference*, pp. 366-374, 1997.

[14] T.S.Papatheodorou and Y.G.Saridakis, "Parallel Algorithms and Architectures for Multisplitting Iterative methods", *Parallel Computing*, vol. 12, pp. 171-182, 1989.

[15] SGI Origin 350 server web page, *http://www.sgi.com/products /remarketed/origin350*.

[16] SUN V240z server web page, *http://www.sun.com/servers/ entry/v240*.

[17] H. A. van der Vorst, "Bi-CGSTAB : A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems", *SIAM J. Sci. Statist. Comput.*, vol. 13, pp. 631-644, 1992.