Parallel Iterative Methods

Svetozar Margenov
CLPP BAS

e-mail *margenov@parallel.bas.bg*

Table Contents

Parallel Performance

To establish the theoretical performance, a simple model for non-overlapped arithmetic and communication times is assumed:

- The execution of M a.o. on one processor takes time

  $$T_a = Mt_a$$

  where t_a is the average unit time to perform one a.o. on one processor.

- The communication time to transfer M data elements from one processor to another is approximated by

  $$T_{com} = \ell(t_s + Mt_c)$$

  where t_s is the start-up time and t_c is the time necessary for each of M elements to be sent, and ℓ is the graph distance between the processors.

The standard expressions for parallel speedup $S(N,p)$, and parallel efficiency $E(N,p)$ are used:

$$S(N,p) = \frac{T(N,1)}{T(N,p)}$$

$$E(N,p) = \frac{S(N,p)}{p}$$

Here, $T(N,p)$ stands for the parallel time to solve the problem on p processors, and N is the discrete size of the problem.
The following theoretical estimates hold:

$$0 < S(N,p) \leq p \qquad 0 < E(N,P) \leq 1$$

Introduction

Iterative methods are techniques to solve systems of linear equations

$$A\widehat{x} = b$$

that generate a sequence of approximations to the solution vector $\widehat{x}$ in the form

$$x^0, x^1, \cdots, x^k, \cdots$$

The process is said to be convergent if the magnitude of the vector

$$r^k = b - Ax^k$$

becomes reasonably small. The vector r^k represents the error in the approximation of $\widehat{x}$ and is referred to as the **residual** after k iterations.

The relative stopping criteria is determined by the inner product of the k^{th} residual

$$\frac{||r^k||_2}{||r^0||_2} < \epsilon$$

where $\epsilon > 0$ is assumed small, and

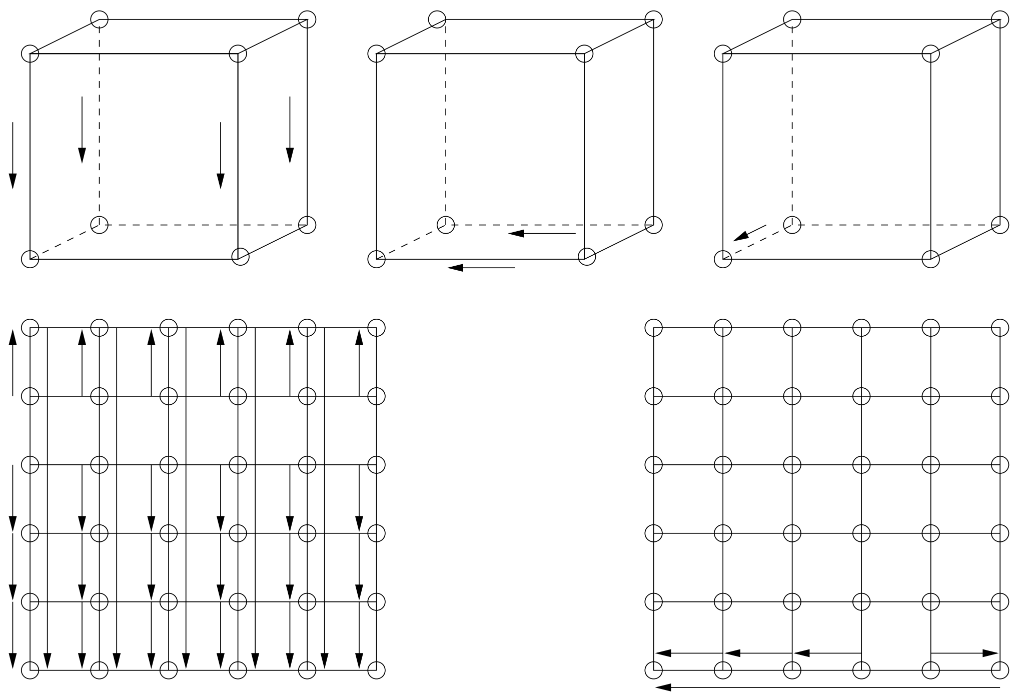
$$||r^i||_2 = \sqrt{{r^i}^T r^i}.$$

In this general setting, each iteration step includes:

- inner product
- matrix A vector multiplication

Parallel Inner Product

The parallel implementation of *inner product* is the only step of the considered iterative algorithms which requires global communications.



$T^{IP}_{com} = 2(t_s + t_c)\log p$ d cube

$T^{IP}_{com} = 4(t_s + t_c)\lceil\sqrt{p}/2\rceil$ 2D mesh

Sparse Matrices

Example A1

$$A = \begin{bmatrix} 4 & & & & & -1 & & & -1 \\ & 4 & & & & & & -1 & -1 \\ & & 4 & & & & -1 & -1 & \\ & & & 4 & & -1 & -1 & & \\ & & & & 4 & -1 & -1 & -1 & -1 \\ -1 & & & -1 & -1 & 4 & & & \\ & & -1 & -1 & -1 & & 4 & & \\ & -1 & -1 & & -1 & & & 4 & \\ -1 & -1 & & & -1 & & & & 4 \end{bmatrix}$$

Example A2

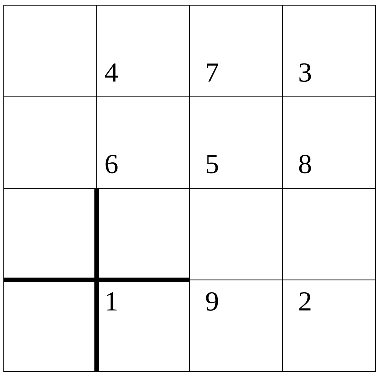
$$A = \begin{bmatrix} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{bmatrix}$$

FDM/FEM Sparse Matrices I

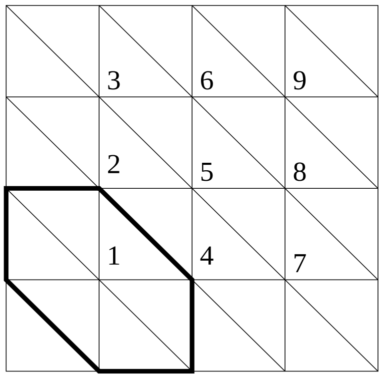
Let us consider the model problem

$$-u_{xx} - u_{yy} = f \qquad in \quad \Omega = [0,1]^2$$

with Dirichlet boundary conditions on $\Gamma = \partial\Omega$. Let us assume that FDM or FEM is used to solve numerically the problem where ω_h is a uniform mesh with mesh-size $h = 1/(n+1)$.



(A1)



(A2)

Let us assume that a column-wise numbering of the nodes (unknowns) is used. Then

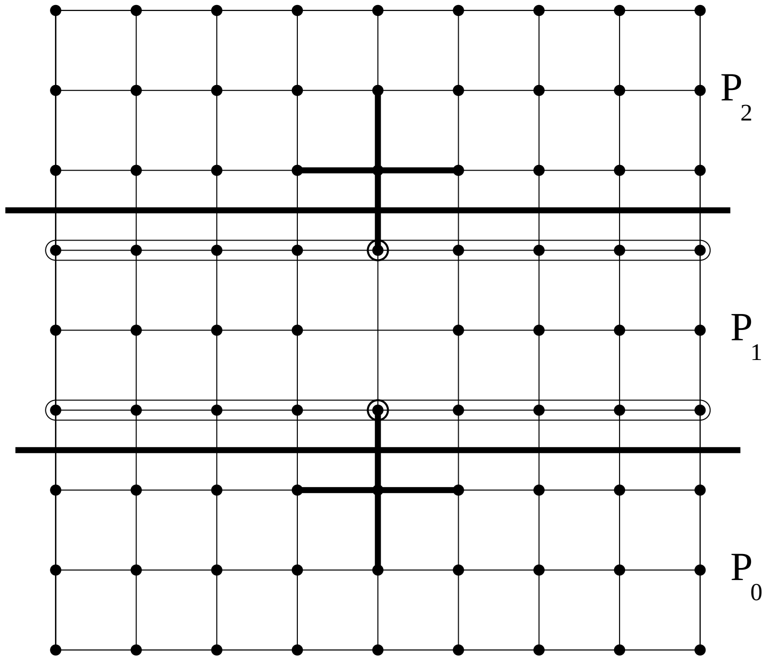
$$A = block - tridiag(A_{i,i-1}, A_{i,i}, A_{i,i+1})$$

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & & & \\ A_{2,1} & A_{2,2} & A_{2,3} & & \\ & A_{3,2} & A_{3,3} & A_{3,4} & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & A_{n-1,n-2} & A_{n-1,n-1} & A_{n-1,n} \\ & & & A_{n,n-1} & A_{n,n} \end{bmatrix}$$

$$A_{i,i} = tridiag(-1, 4, -1), \quad A_{i,i-1} = A_{i,i+1} = -I$$

$$A \in \mathcal{R}^{N\times N}, \quad A_{i,j} \in \mathcal{R}^{n\times n}, \quad N = n^2$$

Matrix-vector multiplication with a block-stripped partitioning of the block-tridiagonal sparse matrix



$$T(N,1) \approx 9Nt_a$$

$$T_{com} = 2(t_s + nt_c)$$

$$T(N,p) \approx \frac{9N}{p}t_a + 2(t_s + \sqrt{N}t_c)$$

Jacobi Iterative Method

The i^{th} equation of the system $A\widehat{x} = b$ can be written in the form

$$\widehat{x}_i = \frac{1}{A_{i,i}} \left(b_i - \sum_{i \neq j} A_{i,j} \widehat{x}_j \right).$$

A typical iteration step in the Jacobi method is

$$x_i^{k+1} = \frac{1}{A_{i,i}} \left(b_i - \sum_{i \neq j} A_{i,j} x_j^k \right)$$

or equivalently

$$x_i^{k+1} = \frac{r_i^k}{A_{i,i}} + x_i^k.$$

One class of matrices for which the method always converges is that of **diagonally-dominant matrices**.

procedure JACOBI_METHOD(A, b, x, ϵ)
begin
    $k := 0$;
    Select initial solution vector x^0;
    $r^0 := b - Ax^0$;
    while ($||r^k||_2 > \epsilon ||r^0||_2$) **do**
    begin
        $k := k + 1$;
        for $i := 1$ **to** N **do**
            $x_i^k := r_i^{k-1}/A_{i,i} + x_i^{k-1}$;
        $r^k := b - Ax^k$;
    endwhile;
    $x := x^k$;
endJACOBI_METHOD

$$T^{it}(N, 1) \approx 14 N t_a$$

$$T^{it}_{com} = 2(t_s + n t_c) + T^{IP}_{com}$$

$$T^{it}(N, p) \approx \frac{14N}{p} t_a + 2(t_s + \sqrt{N} t_c)$$

Conjugate Gradient Method

$$x^0, \quad g^0 = Ax^0 - b, \quad d^0 = -g^0,$$

$$k = 0, 1, 2, \ldots$$

$$\tau_k = \frac{{g^k}^T g^k}{{d^k}^T A d^k}$$

$$x^{k+1} = x^k + \tau_k d^k$$

$$g^{k+1} = g^k + \tau_k A d^k$$

$$\beta_k = \frac{{g^{k+1}}^T g^{k+1}}{{g^k}^T g^k}$$

$$d^{k+1} = -g^{k+1} + \beta_k d^k$$

The computational complexity of one CG iteration is as follows:

$$\mathcal{N}_{it}^{CG}(A^{-1}b) \approx \mathcal{N}(Ad) + 2\mathcal{N}(IP) + 3\mathcal{N}(LT)$$

$$\mathcal{N}_{it}^{CG}(A^{-1}b) \approx 19N$$

The related times are derived in a similar way as for Jacobi method.

$$T^{it}(N,1) \approx 19Nt_a$$

$$T_{com}^{it} = 2(t_s + nt_c) + 2T_{com}^{IP}$$

$$T^{it}(N,p) \approx \frac{19N}{p}t_a + 2(t_s + \sqrt{N}t_c)$$

Convergence rate of CG method

Theorem.

$$p(\epsilon) \leq \frac{1}{2}\sqrt{\kappa(A)}\,\ln(2/\epsilon) + 1,$$

where $p(\epsilon)$ stands for the smallest number k such that

$$||x^k - \widehat{x}||_A \leq \epsilon ||x^0 - \widehat{x}||_A \qquad \forall x^0 \in \mathcal{R}^N$$

In a very general setting of FEM/FDM sparse matrices, the condition number behaves as

$$\kappa(A) = O(N),$$

and therefore

$$\mathcal{N}^{CG}(A^{-1}b) = O(N^{3/2}).$$

Remark. The same complexity holds for the best known direct method, namely

$$\mathcal{N}^{ND}(A^{-1}b) = O(N^{3/2})$$

where ND stands for the *Nested Dissection Method*.

Numerical Tests

n	$G-S$	SD	CG	PCG
4	82	185	26	11
8	309	698	45	15
16	1151	2592	91	19
32	4242	9541	177	27
64	15529	34818	351	38

$$n_{it}^{G-S} = O(N) \qquad n_{it}^{SD} = O(N)$$
$$n_{it}^{CG} = O(N^{1/2}) \quad n_{it}^{PCG} = O(N^{1/4})$$

Conclusion. The efforts should be addressed to development of scalable parallel algorithms for *fast enough* iterative solution methods.

Preconditioned Conjugate Gradient Method

The idea of the PCG method is to substitute the original linear system to a new one which is **better conditioned**:

$$A\hat{x} = b \quad => \quad C^{-1/2} A C^{-1/2} \hat{y} = \bar{b}$$

The PCG strategy is to construct a preconditioner C such that:

- $\kappa(C^{-1}A) << \kappa(A)$
- $\mathcal{N}(C^{-1}v) << \mathcal{N}(A^{-1}v)$

The preconditioner is called optimal if $\kappa(C^{-1}A) = O(1)$ and $\mathcal{N}(C^{-1}v) = O(N)$.

PCG algorithm

$$x^0, \quad g^0 = Ax^0 - b, \quad h^0 = C^{-1}g^0, \quad d^0 = h^0$$

$$k = 0, 1, 2, \ldots$$

$$\tau_k = \frac{g^{k^T} h^k}{d^{k^T} A d^k}$$

$$x^{k+1} = x^k + \tau_k d^k$$

$$g^{k+1} = g^k + \tau_k A d^k$$

$$h^{k+1} = C^{-1} g^{k+1}$$

$$\beta_k = \frac{g^{k+1^T} h^{k+1}}{g^{k^T} h^k}$$

$$d^{k+1} = -h^{k+1} + \beta_k d^k$$

Computational Complexity of One PCG Iteration

$$\mathcal{N}_{it}^{PCG}(A^{-1}b) \approx \mathcal{N}(C^{-1}g) + \mathcal{N}(Ad) + 2\mathcal{N}(IP) + 3\mathcal{N}(LT)$$

$$\mathcal{N}_{it}^{PCG}(A^{-1}b) \approx \mathcal{N}(C^{-1}g) + 19N$$

PCG Times Per Iteration

$$T^{it}(N,1) \approx T^{(C^{-1}g)}(N,1) + 19Nt_a$$

$$T_{com}^{it} = 2(t_s + nt_c) + T_{com}^{(C^{-1}g)} + 2T_{com}^{IP}$$

Convergence rate of PCG method

Theorem.

$$p(\epsilon) \leq \frac{1}{2}\sqrt{\kappa(C^{-1}A)}\ln(2/\epsilon) + 1,$$

where $p(\epsilon)$ stands for the smallest number k such that

$$||x^k - \widehat{x}||_A \leq \epsilon ||x^0 - \widehat{x}||_A \qquad \forall x^0 \in \mathcal{R}^N$$

Preconditioning Techniques:

- Incomplete Factorization
- Domain Decomposition
- Patched Local Refinement
- Multigrid/Multilevel
- Approximate Inverse

MIC(0) Algorithm

Let us rewrite the real matrix A in the form $A = D - L - L^t$.

$$C_{MIC(0)}(A) = (X - L)X^{-1}(X - L)^t$$

where $X = diag(x_1, \cdots, x_N)$ is determined by the condition of equal rowsums.

Theorem. Let us assume that

$$\begin{array}{rcl} L & \geq & 0 \\ A\underline{e} & \geq & 0 \\ A\underline{e} + L^t\underline{e} & > & 0 \qquad \underline{e} = (1, \cdots, 1)^t \in \mathcal{R}^N. \end{array}$$

Then the relation

$$x_i = a_{ii} - \sum_{k=1}^{i-1} \frac{a_{ik}}{x_k} \sum_{j=k+1}^{N} a_{kj}$$

gives the positive values x_i and the diagonal matrix

$$X = diag(x_1, \cdots, x_N)$$

defines stable $MIC(0)$ factorization of A.

Remark. The presented numerical tests are performed using the perturbed $MIC(0)$ algorithm, where the incomplete factorization is applied to the matrix $\tilde{A} = A + \tilde{D}$. The diagonal perturbation $\tilde{D} = \tilde{D}(\xi) = diag(\tilde{d}_1, \dots \tilde{d}_N)$ is defined as follows:

$$\tilde{d}_i = \begin{cases} \xi a_{ii} & if \quad a_{ii} \geq 2w_i \\ \xi^{1/2} a_{ii} & if \quad a_{ii} < 2w_i \end{cases}$$

where

$$w_i = -\sum_{j>i} a_{ij}.$$

Here $0 < \xi < 1$ is a constant of the same order as the minimal eigenvalue of A. The computations for the considerd model problems are done with $\xi = h^2$.

MIC(0) Computational Complexity of One PCG Iteration

$$\mathcal{N}_{it}^{PCG}(A^{-1}b) \approx \mathcal{N}(C^{-1}g) + 19N$$

$$\mathcal{N}(C^{-1}g) \approx 11N$$

$$\mathcal{N}_{it}^{PCG}(A^{-1}b) \approx 30N$$

MIC(0) is a cheap preconditioning algorithm. The cost of $\mathcal{N}(C^{-1}g)$ is almost the same as $\mathcal{N}(Ad)$!

MIC(0) is a robust preconditioner with respect to local singularities of the problem.

$$\kappa(C^{-1}A) = O(N^{1/4}) \quad \mathcal{N}^{PCG}(A^{-1}b) = O(N^{5/4})$$

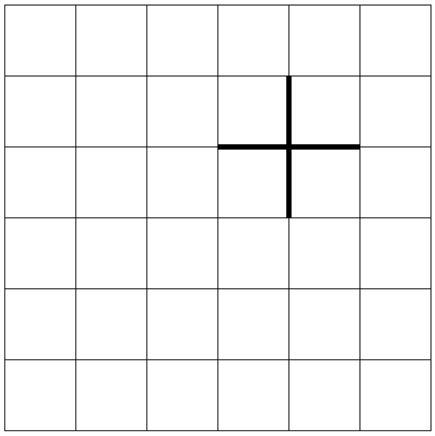
MIC(0) is an inherently sequential algorithm!

FDM/FEM Sparse Matrices II

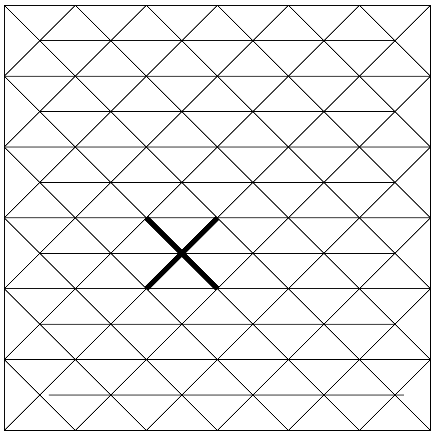
Consider the the model problem

$$-u_{xx} - u_{yy} = f \qquad in \;\; \Omega = [0,1]^2$$

with Dirichlet boundary conditions on $\Gamma = \partial\Omega$.

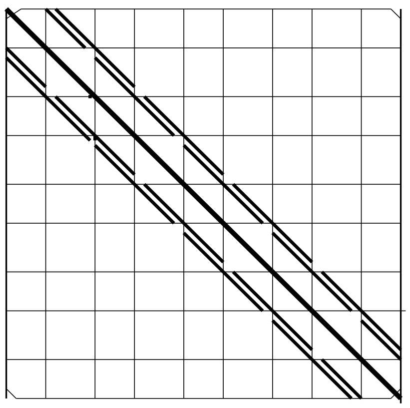


ReM

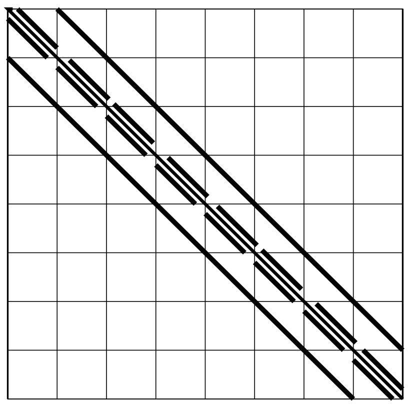


SkM

The approximation properties of the regular mesh (ReM) and the alternative skewed mesh (SkM) are one and the same. A five point stencil is obtained in both cases.



SkM

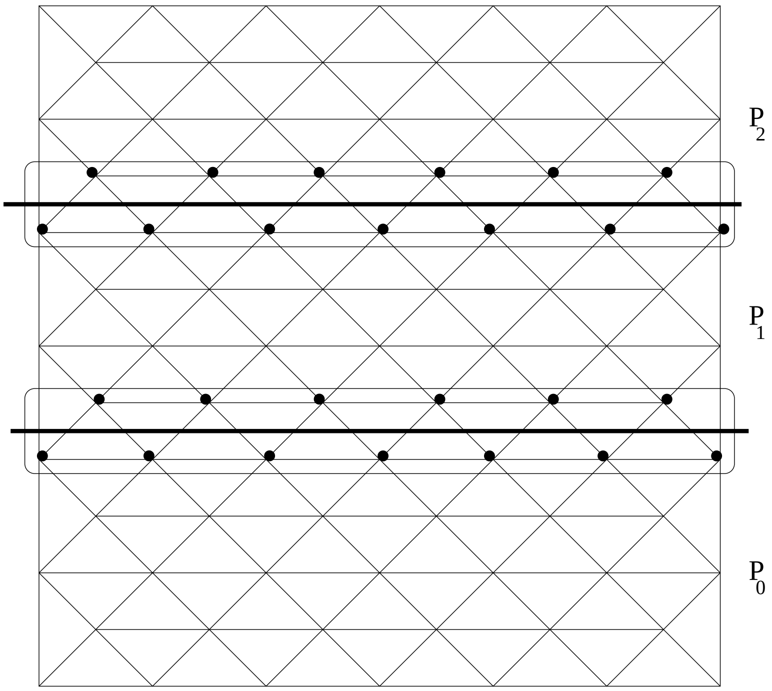


ReM

The bottleneck problem of the parallel implementation of MIC(0) algorithm is the solution of problems with triangle matrices $(X-L)$ and $(X-L)^T$.

The key point of our consideration is, that in the case of skewed mesh, the stiffness matrix has a block structure with diagonal blocks which are diagonal.

MIC(0) PCG algorithm with a block-stripped partitioning



$$N = n^2 + (n-1)^2$$

$$T_{it}^{MIC(0)}(N,1) \approx 38Nt_a$$

$$T_{com} \approx (4t_s + 6t_c)n + 2T_{com}^{IP}$$

$$T_{it}^{MIC(0)}(N,p) \approx \frac{38N}{p}t_a + (2t_s + 3t_c)\sqrt{2N}$$

MIC(0) PCG: Parallel Numerical Tests

The presented numerical tests are performed on a Beowulf like cluster of four dual processor Power Macintosh computers with 512 MB RAM each and G4 processors on 450 MHz.

The parallel MIC(0) algorithm is implemented in C++ using Message Passing Interface (MPI).

Yellow Dog Linux with LAN MPI are used.

The size of the problem and the number of the processors are varied to examine the parallel scalability of the code.

Parallel Speedup

n	32	64	128	256
S(n,2)	1.21	1.68	1.96	1.85
S(n,3)	0.24	0.46	0.97	1.72
S(n,4)	0.22	0.46	1.11	1.97
S(n,5)	0.20	0.40	0.96	1.99
S(n,6)	0.18	0.39	1.03	1.99
S(n,7)	0.19	0.38	0.95	1.78
S(n,8)	0.19	0.39	1.00	2.28
n	512	1024	1500	
S(n,2)	1.92	2.03	2.02	
S(n,3)	2.45	2.97	2.86	
S(n,4)	2.88	3.76	3.95	
S(n,5)	3.25	4.48	4.86	
S(n,6)	3.55	5.23	5.73	
S(n,7)	3.63	6.02	6.31	
S(n,8)	3.97	6.37	6.76	

Parallel Efficiency

n	32	64	128	256
E(n,2)	0.60	0.84	0.98	0.93
E(n,3)	0.08	0.15	0.32	0.57
E(n,4)	0.06	0.12	0.27	0.49
E(n,5)	0.04	0.08	0.19	0.40
E(n,6)	0.03	0.07	0.17	0.33
E(n,7)	0.03	0.05	0.14	0.25
E(n,8)	0.02	0.05	0.13	0.29
n	512	1024	1500	
E(n,2)	0.96	1.02	1.01	
E(n,3)	0.81	0.99	0.95	
E(n,4)	0.72	0.94	0.98	
E(n,5)	0.65	0.90	0.97	
E(n,6)	0.59	0.87	0.96	
E(n,7)	0.51	0.86	0.90	
E(n,8)	0.50	0.80	0.84	