# Parallel Performance of a 3D Elliptic Solver

Ivan Lirkov

Central Laboratory for Parallel Processing, Bulgarian Academy of Sciences,
Acad.G.Bonchev, Bl. 25A,
1113 Sofia, Bulgaria
ivan@parallel.bas.bg

**Abstract.** It was recently shown that block-circulant preconditioners applied to a conjugate gradient method used to solve structured sparse linear systems arising from 2D or 3D elliptic problems have good numerical properties and a potential for high parallel efficiency. The asymptotic estimate for their convergence rate is as for the incomplete factorization methods but the efficiency of the parallel algorithms based on circulant preconditioners are asymptotically optimal. In this paper parallel performance of a circulant block-factorization based preconditioner applied to a 3D model problem is investigated. The aim of this presentation is to analyze the performance and to report on the experimental results obtained on shared and distributed memory parallel architectures. A portable parallel code is developed based on Message Passing Interface (MPI) and OpenMP (Open Multi Processing) standards. The performed numerical tests on a wide range of parallel computer systems clearly demonstrate the high level of parallel efficiency of the developed parallel code.

## 1 Introduction

In this article we are concerned with the numerical solution of 3D linear boundary value problems of elliptic type. After discretization, such problems lead to find the solution of linear systems of the form $A\mathbf{x} = \mathbf{b}$. We shall only consider the case where $A$ is symmetric and positive definite. In practice, large problems of this class are often solved by iterative methods, such as the conjugate gradient method. At each step of these iterative methods only the product of $A$ with a given vector $\mathbf{v}$ is needed. Such methods are therefore ideally suited to exploit the sparsity of $A$.

Typically, the rate of convergence of these methods depends on the condition number $\kappa(A)$ of the coefficient matrix $A$: the smaller $\kappa(A)$ leads to the faster convergence. Unfortunately, for elliptic problems of second order, usually $\kappa(A) = \mathcal{O}(n^2)$, where $n$ is the number of mesh points in each coordinate direction, and hence grows rapidly with $n$. To somewhat facilitate this problem, these methods are almost always used with a preconditioner $M$. The preconditioner is chosen with two criteria in mind: to minimize $\kappa(M^{-1}A)$ and to allow efficient computation of the product $M^{-1}\mathbf{v}$ for a given vector $\mathbf{v}$. These two goals are often conflicting ones and much research has been done into devising preconditioners

Z. Li et al. (Eds.): NAA 2004, LNCS 3401, pp. 383–390, 2005.

that strike a delicate balance between both. Recently, a third aspect has been added to the above two, namely, the possibility to easily implement the action of the preconditioner on a parallel computer system.

One of the most popular and the most successful class of preconditioners is the class of incomplete $LU$ (ILU) factorizations. One potential problem with the ILU preconditioners is that they have limited degree of parallelism. Some attempts to modify the method and to devise more parallel variants often result in a deterioration of the convergence rate.

R. Chan and T. F. Chan [1] proposed another class of preconditioners which is based on averaging coefficients of $A$ to form a block-circulant approximation. The block-circulant preconditioners are highly parallelizable but they are substantially sensitive with respect to a possible high variation of the coefficients of the elliptic operator.

The sensitivity of the block-circulant approximations with respect to a high variation of the problem coefficients was relaxed in the circulant block factorization (CBF) preconditioners [4].

The main goal of this study is analysis of the parallel complexity of the PCG method with considered circulant block-factorization preconditioners obtained on Cray T3E-900, SUNfire 6800, and NEC server Azusa Express5800/1160Xa computers, Linux Athlon, Macintosh, and Cray Opteron clusters.

## 2 Circulant Block Factorization

Let us recall that an $m \times m$ circulant matrix $C$ has the form $(C_{k,j}) = \left(c_{(j-k) \bmod m}\right)$. Any circulant matrix can be factorized as $C = F \Lambda F^*$ where $\Lambda$ is a diagonal matrix containing the eigenvalues of $C$, and $F$ is the Fourier matrix $F = \frac{1}{\sqrt{m}} \left\{ e^{2\pi \frac{jk}{m} \mathrm{i}} \right\}_{0 \le j,k \le m-1}$. Here i stands for the imaginary unit. $F^* = \overline{F}^T$ denotes adjoint matrix of $F$.

The CBF preconditioning technique incorporates the circulant approximations into the framework of the $LU$ block factorization. The computational efficiency and parallelization of the resulting algorithm are as high as of the block circulant one (see [1, 3]).

The following 3D elliptic problem is considered in [2]:

$$-(a(x_1,x_2,x_3)u_{x_1})_{x_1} - (b(x_1,x_2,x_3)u_{x_2})_{x_2} - (c(x_1,x_2,x_3)u_{x_3})_{x_3} = f(x_1,x_2,x_3)$$

on the unit cube $[0,1] \times [0,1] \times [0,1]$ with Dirichlet boundary condition. If the domain is discretized by uniform grid with $n_1$, $n_2$ and $n_3$ grid points along the coordinate directions, and if a standard (for such a problem) seven-point FDM (FEM) approximation is used, then the stiffness matrix $A$ admits a block-tridiagonal structure. The matrix $A$ can be written in the block-form

$$A = tridiag(A_{i,i-1}, A_{i,i}, A_{i,i+1}) \qquad i = 1, 2, \ldots, n_1,$$

where $A_{i,i}$ are block-tridiagonal matrices which correspond to one $x_1$-plane and off-diagonal blocks are diagonal matrices.

In this case the general CBF preconditioning approach (see [4]) is applied to construct the preconditioner $M_{CBF}$ in the form:

$$M_{CBF} = tridiag(C_{i,i-1}, C_{i,i}, C_{i,i+1}) \qquad i = 1, 2, \ldots n_1, \tag{1}$$

where $C_{i,j} = Block - Circulant(A_{i,j})$ is a block-circulant approximation of the corresponding block $A_{i,j}$. The relative condition number of the CBF preconditioner for the model (Laplace) 3D problem is analyzed using the technique from [5] and the following estimate is derived:

$$\kappa(M_0^{-1} A_0) \leq 2 \max(n_2, n_3) + 2\sqrt{2}. \tag{2}$$

An algorithm to construct a preconditioner $M$ for a given block-tridiagonal matrix was described above. The main advantage of the circulant preconditioners is that they possess much more parallelism compared to the ILU preconditioners. It is described in [3] how to implement an action of the inverse of this preconditioner on a given vector. For our preconditioner, $\lim_{n\to\infty} S_p = p$ and $\lim_{n\to\infty} E_p = 1$, (see [3]) i.e., the algorithm is asymptotically optimal.

## 3 Experimental Results

In this section we report the results of the experiments executed on a NEC server Azusa Express5800/1160Xa consisting of 16 x intel Itanum 800Mhz processors, with 32 GB main memory (see http://www.hlrs.de/hw-access/platforms/azusa/); a Cray T3E-900 consisting of 336 Digital Alpha 450 MHz processors, with 64 or 128 MB memory on processor; a SUNfire 6800 consisting of 24 UltraSPARC-III 750 MHz processors and 48 GB main memory; and Linux clusters consisting of: 256 AMD Opteron 2 GHz processors, 516 GB memory (see http://www.hlrs.de/hw-access/platforms/strider/); 17 PC with AMD Athlon 650 MHz processors, 128 MB memory per computer, and 4 dual processor PowerPC with G4 450 MHz processors, 512 MB memory per node. The developed parallel code has been implemented in C and the parallelization has been facilitated using the MPI [8, 9] and OpenMP [10] libraries. In all cases, the optimization options of the compiler have been tuned to achieve the best performance. Times have been collected using the MPI provided timer. In all cases we report the best results from multiple runs.

The obtained parallel time $T_p$ on $p$ processors, relative parallel speed-up $S_p$ and relative efficiency $E_p$ are presented in the tables, where $S_p = \frac{T_1}{T_p} \leq p$ and $E_p = \frac{S_p}{p} \leq 1$. One can see the increase of the influence of the communication time with the number of processors on the speed-up and on the parallel efficiency. The general behavior is in a good agreement with the theoretical estimates.

In Table 1 we present results of experiments executed on Athlon and Macintosh clusters, on Cray T3E, and on NEC server Azusa Express5800/1160Xa. One can see that the parallel efficiency on Macintosh cluster is higher on 2 processors and it is lower on 6 and 8 processors. The main reason is the faster communication between processors in one node on the cluster of dual processor computers.

**Table 1.** Parallel time, speed-up and parallel efficiency for the CBF preconditioner on Athlon and Macintosh clusters, Cray T3E, and NEC server Azusa Express5800/1160Xa

| | | Athlon | | | Macintosh | | | Cray T3E | | | NEC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | p | $T_p$ | $S_p$ | $E_p$ | $T_p$ | $S_p$ | $E_p$ | $T_p$ | $S_p$ | $E_p$ | $T_p$ | $S_p$ | $E_p$ |
| 32 | 1 | 0.111 | | | 0.109 | | | 0.133 | | | 0.077 | | |
| | 2 | 0.082 | 1.35 | 0.673 | 0.061 | 1.78 | 0.891 | 0.068 | 1.96 | 0.982 | 0.040 | 1.95 | 0.975 |
| | 4 | 0.057 | 1.95 | 0.487 | 0.050 | 2.18 | 0.544 | 0.034 | 3.88 | 0.969 | 0.020 | 3.86 | 0.965 |
| | 8 | 0.031 | 3.61 | 0.452 | 0.036 | 3.05 | 0.382 | 0.020 | 6.64 | 0.830 | 0.011 | 7.23 | 0.904 |
| | 16 | | | | | | | 0.014 | 9.61 | 0.601 | | | |
| | 32 | | | | | | | 0.011 | 12.58 | 0.393 | | | |
| 48 | 1 | 0.758 | | | 0.761 | | | 0.898 | | | 0.470 | | |
| | 2 | 0.469 | 1.62 | 0.809 | 0.405 | 1.88 | 0.939 | 0.452 | 1.99 | 0.993 | 0.234 | 2.01 | 1.004 |
| | 3 | 0.340 | 2.23 | 0.744 | 0.318 | 2.39 | 0.798 | 0.304 | 2.95 | 0.984 | 0.156 | 3.01 | 1.004 |
| | 4 | 0.261 | 2.91 | 0.726 | 0.265 | 2.87 | 0.718 | 0.227 | 3.95 | 0.988 | 0.117 | 4.01 | 1.002 |
| | 6 | 0.189 | 4.01 | 0.669 | 0.200 | 3.81 | 0.634 | 0.154 | 5.82 | 0.971 | 0.079 | 5.94 | 0.989 |
| | 8 | 0.203 | 3.74 | 0.468 | 0.164 | 4.63 | 0.579 | 0.116 | 7.71 | 0.964 | 0.060 | 7.78 | 0.973 |
| | 12 | | | | | | | 0.080 | 11.29 | 0.941 | 0.042 | 11.30 | 0.942 |
| | 16 | | | | | | | 0.062 | 14.58 | 0.911 | | | |
| | 24 | | | | | | | 0.044 | 20.27 | 0.845 | | | |
| | 48 | | | | | | | 0.027 | 33.84 | 0.705 | | | |
| 64 | 1 | 1.095 | | | 1.191 | | | | | | 0.744 | | |
| | 2 | 0.753 | 1.45 | 0.727 | 0.705 | 1.69 | 0.845 | 0.583 | | | 0.364 | 2.05 | 1.023 |
| | 4 | 0.429 | 2.55 | 0.638 | 0.495 | 2.41 | 0.602 | 0.297 | | 0.981 | 0.177 | 4.20 | 1.051 |
| | 8 | 0.268 | 4.08 | 0.511 | 0.327 | 3.64 | 0.456 | 0.153 | | 0.953 | 0.092 | 8.06 | 1.007 |
| | 16 | | | | | | | 0.079 | | 0.922 | | | |
| | 32 | | | | | | | 0.045 | | 0.810 | | | |
| | 64 | | | | | | | 0.028 | | 0.651 | | | |
| 96 | 1 | 6.787 | | | 6.701 | | | | | | 4.332 | | |
| | 2 | 4.107 | 1.65 | 0.826 | 3.748 | 1.79 | 0.894 | | | | 2.185 | 1.98 | 0.991 |
| | 3 | 2.975 | 2.28 | 0.760 | 2.871 | 2.33 | 0.778 | | | | 1.467 | 2.95 | 0.984 |
| | 4 | 2.202 | 3.08 | 0.770 | 2.398 | 2.79 | 0.699 | | | | 1.083 | 4.00 | 1.000 |
| | 6 | 1.530 | 4.44 | 0.739 | 1.743 | 3.84 | 0.641 | 1.256 | | | 0.733 | 5.91 | 0.985 |
| | 8 | 1.160 | 5.85 | 0.731 | 1.374 | 4.88 | 0.610 | 0.944 | | 0.998 | 0.561 | 7.73 | 0.966 |
| | 12 | | | | | | | 0.639 | | 0.983 | 0.397 | 10.91 | 0.909 |
| | 16 | | | | | | | 0.484 | | 0.973 | | | |
| | 24 | | | | | | | 0.322 | | 0.975 | | | |
| | 32 | | | | | | | 0.243 | | 0.969 | | | |
| | 48 | | | | | | | 0.170 | | 0.924 | | | |
| | 96 | | | | | | | 0.095 | | 0.826 | | | |
| 128 | 1 | | | | | | | | | | 7.240 | | |
| | 2 | | | | | | | | | | 3.576 | 2.02 | 1.012 |
| | 4 | | | | | | | | | | 1.897 | 3.82 | 0.954 |
| | 8 | | | | | | | | | | 1.045 | 6.93 | 0.866 |
| 192 | 1 | | | | | | | | | | 37.901 | | |
| | 2 | | | | | | | | | | 18.096 | 2.09 | 1.047 |
| | 3 | | | | | | | | | | 12.280 | 3.09 | 1.029 |
| | 4 | | | | | | | | | | 9.345 | 4.06 | 1.014 |
| | 6 | | | | | | | | | | 6.627 | 5.72 | 0.953 |
| | 8 | | | | | | | | | | 5.348 | 7.09 | 0.886 |
| | 12 | | | | | | | | | | 4.082 | 9.28 | 0.774 |
| 256 | 1 | | | | | | | | | | 64.039 | | |
| | 2 | | | | | | | | | | 30.626 | 2.09 | 1.045 |
| | 4 | | | | | | | | | | 15.636 | 4.10 | 1.024 |
| | 8 | | | | | | | | | | 9.437 | 6.79 | 0.848 |

**Table 2.** Parallel time, speed-up and parallel efficiency for the CBF preconditioner on SUNfire 6800 and on Cray Opteron cluster using MPI

| | SUNfire | | | | | | | | Cray Opteron | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| p | n | $T_p$ | $S_p$ | $E_p$ | n | $T_p$ | $S_p$ | $E_p$ | n | $T_p$ | $S_p$ | $E_p$ | n | $T_p$ | $S_p$ | $E_p$ |
| 1 | 32 | 0.050 | | | 128 | 8.445 | | | 32 | 0.034 | | | 128 | 2.768 | | |
| 2 | | 0.034 | 1.48 | 0.741 | | 4.386 | 1.93 | 0.963 | | 0.017 | 2.06 | 1.030 | | 1.428 | 1.94 | 0.969 |
| 4 | | 0.015 | 3.27 | 0.818 | | 2.105 | 4.01 | 1.003 | | 0.009 | 3.94 | 0.985 | | 0.750 | 3.69 | 0.923 |
| 8 | | 0.009 | 5.84 | 0.729 | | 0.898 | 9.40 | 1.175 | | 0.005 | 7.16 | 0.895 | | 0.402 | 6.88 | 0.861 |
| 16 | | 0.004 | 11.28 | 0.705 | | 0.386 | 21.88 | 1.367 | | 0.003 | 12.07 | 0.754 | | 0.222 | 12.47 | 0.779 |
| 32 | | | | | | | | | | 0.003 | 13.17 | 0.412 | | 0.127 | 21.80 | 0.681 |
| 64 | | | | | | | | | | | | | | 0.068 | 40.67 | 0.635 |
| 128 | | | | | | | | | | | | | | 0.045 | 61.22 | 0.478 |
| 1 | 48 | 0.404 | | | 192 | 41.063 | | | 48 | 0.232 | | | 192 | 16.576 | | |
| 2 | | 0.242 | 1.67 | 0.834 | | 20.448 | 2.01 | 1.004 | | 0.116 | 2.00 | 0.998 | | 8.591 | 1.93 | 0.965 |
| 3 | | 0.146 | 2.77 | 0.924 | | 14.141 | 2.90 | 0.968 | | 0.079 | 2.95 | 0.984 | | 5.715 | 2.90 | 0.967 |
| 4 | | 0.111 | 3.65 | 0.912 | | 10.279 | 3.99 | 0.999 | | 0.059 | 3.96 | 0.989 | | 4.391 | 3.78 | 0.944 |
| 6 | | 0.068 | 5.97 | 0.995 | | 7.004 | 5.86 | 0.977 | | 0.040 | 5.74 | 0.957 | | 3.023 | 5.48 | 0.914 |
| 8 | | 0.053 | 7.62 | 0.953 | | 5.155 | 7.96 | 0.996 | | 0.031 | 7.44 | 0.930 | | 2.258 | 7.34 | 0.918 |
| 12 | | | | | | | | | | 0.021 | 11.26 | 0.938 | | 1.533 | 10.81 | 0.901 |
| 16 | | 0.024 | 16.73 | 1.045 | | 2.346 | 17.50 | 1.094 | | 0.016 | 14.67 | 0.917 | | 1.188 | 13.95 | 0.872 |
| 24 | | 0.019 | 21.65 | 0.902 | | 1.737 | 23.63 | 0.985 | | 0.011 | 20.66 | 0.861 | | 0.816 | 20.31 | 0.846 |
| 32 | | | | | | | | | | | | | | 0.633 | 26.17 | 0.818 |
| 48 | | | | | | | | | | 0.007 | 31.80 | 0.662 | | 0.445 | 37.27 | 0.776 |
| 64 | | | | | | | | | | | | | | 0.344 | 48.14 | 0.752 |
| 96 | | | | | | | | | | | | | | 0.246 | 67.38 | 0.702 |
| 192 | | | | | | | | | | | | | | 0.139 | 118.99 | 0.620 |
| 1 | 64 | 0.672 | | | 256 | | | | 64 | 0.307 | | | 256 | 24.442 | | |
| 2 | | 0.388 | 1.73 | 0.866 | | 38.417 | | | | 0.162 | 1.90 | 0.950 | | 13.104 | 1.87 | 0.933 |
| 4 | | 0.160 | 4.20 | 1.050 | | 18.642 | | 1.030 | | 0.084 | 3.66 | 0.914 | | 6.896 | 3.54 | 0.886 |
| 8 | | 0.075 | 8.97 | 1.122 | | 9.826 | | 0.977 | | 0.044 | 7.00 | 0.875 | | 3.510 | 6.96 | 0.870 |
| 16 | | 0.031 | 22.01 | 1.375 | | 4.624 | | 1.039 | | 0.026 | 11.91 | 0.744 | | 1.915 | 12.76 | 0.798 |
| 32 | | | | | | | | | | 0.013 | 23.68 | 0.740 | | 1.078 | 22.68 | 0.709 |
| 64 | | | | | | | | | | 0.009 | 35.57 | 0.556 | | 0.587 | 41.62 | 0.650 |
| 128 | | | | | | | | | | | | | | 0.356 | 68.65 | 0.536 |
| 1 | 96 | 4.695 | | | | | | | 96 | 1.973 | | | | | | |
| 2 | | 2.302 | 2.04 | 1.020 | | | | | | 1.010 | 1.95 | 0.977 | | | | |
| 3 | | 1.499 | 3.13 | 1.044 | | | | | | 0.688 | 2.87 | 0.956 | | | | |
| 4 | | 1.091 | 4.30 | 1.076 | | | | | | 0.516 | 3.82 | 0.956 | | | | |
| 6 | | 0.682 | 6.88 | 1.147 | | | | | | 0.361 | 5.47 | 0.912 | | | | |
| 8 | | 0.494 | 9.49 | 1.187 | | | | | | 0.272 | 7.26 | 0.907 | | | | |
| 12 | | | | | | | | | | 0.184 | 10.73 | 0.894 | | | | |
| 16 | | 0.242 | 19.43 | 1.214 | | | | | | 0.145 | 13.63 | 0.852 | | | | |
| 24 | | 0.191 | 24.63 | 1.026 | | | | | | 0.100 | 19.79 | 0.825 | | | | |
| 32 | | | | | | | | | | 0.076 | 25.87 | 0.808 | | | | |
| 48 | | | | | | | | | | 0.053 | 36.97 | 0.770 | | | | |
| 96 | | | | | | | | | | 0.031 | 64.58 | 0.673 | | | | |

**Table 3.** Parallel time, speed-up and parallel efficiency for the CBF preconditioner on SUNfire 6800 using OpenMP

| p | n | $T_p$ | $S_p$ | $E_p$ | n | $T_p$ | $S_p$ | $E_p$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 32 | 0.056 | | | 128 | 8.313 | | |
| 2 | | 0.028 | 1.97 | 0.983 | | 4.233 | 1.96 | 0.982 |
| 3 | | 0.020 | 2.75 | 0.918 | | 2.757 | 3.01 | 1.005 |
| 4 | | 0.015 | 3.69 | 0.922 | | 2.029 | 4.10 | 1.024 |
| 5 | | 0.014 | 4.10 | 0.820 | | 1.633 | 5.09 | 1.018 |
| 6 | | 0.012 | 4.65 | 0.775 | | 1.387 | 5.99 | 0.999 |
| 7 | | 0.011 | 5.11 | 0.730 | | 1.196 | 6.95 | 0.993 |
| 8 | | 0.009 | 6.43 | 0.803 | | 1.039 | 8.00 | 1.000 |
| 16 | | 0.006 | 9.71 | 0.607 | | 0.538 | 15.44 | 0.965 |
| 24 | | 0.006 | 9.70 | 0.404 | | 0.398 | 20.89 | 0.870 |
| 1 | 48 | 0.386 | | | 192 | 40.471 | | |
| 2 | | 0.196 | 1.97 | 0.984 | | 20.181 | 2.01 | 1.003 |
| 3 | | 0.130 | 2.96 | 0.988 | | 13.437 | 3.01 | 1.004 |
| 4 | | 0.099 | 3.91 | 0.977 | | 10.048 | 4.03 | 1.007 |
| 5 | | 0.082 | 4.70 | 0.941 | | 8.044 | 5.03 | 1.006 |
| 6 | | 0.067 | 5.79 | 0.964 | | 6.751 | 6.00 | 0.999 |
| 7 | | 0.060 | 6.48 | 0.926 | | 5.808 | 6.97 | 0.996 |
| 8 | | 0.052 | 7.40 | 0.925 | | 5.086 | 7.96 | 0.995 |
| 16 | | 0.029 | 13.42 | 0.838 | | 2.514 | 16.10 | 1.006 |
| 24 | | 0.020 | 19.38 | 0.807 | | 1.885 | 21.47 | 0.894 |
| 1 | 64 | 0.650 | | | 256 | | | |
| 2 | | 0.339 | 1.92 | 0.958 | | 51.349 | | |
| 3 | | 0.230 | 2.82 | 0.940 | | 33.793 | | 1.013 |
| 4 | | 0.170 | 3.82 | 0.954 | | 25.664 | | 1.000 |
| 5 | | 0.140 | 4.65 | 0.930 | | 20.521 | | 1.001 |
| 6 | | 0.119 | 5.46 | 0.910 | | 17.093 | | 1.001 |
| 7 | | 0.104 | 6.27 | 0.896 | | 14.707 | | 0.998 |
| 8 | | 0.087 | 7.46 | 0.933 | | 13.242 | | 0.969 |
| 16 | | 0.049 | 13.23 | 0.827 | | 6.918 | | 0.928 |
| 24 | | 0.040 | 16.40 | 0.683 | | 5.116 | | 0.836 |
| 1 | 96 | 3.973 | | | | | | |
| 2 | | 1.880 | 2.11 | 1.057 | | | | |
| 3 | | 1.224 | 3.25 | 1.082 | | | | |
| 4 | | 0.930 | 4.27 | 1.068 | | | | |
| 5 | | 0.759 | 5.24 | 1.048 | | | | |
| 6 | | 0.622 | 6.39 | 1.065 | | | | |
| 7 | | 0.542 | 7.33 | 1.047 | | | | |
| 8 | | 0.471 | 8.44 | 1.055 | | | | |
| 16 | | 0.245 | 16.23 | 1.015 | | | | |
| 24 | | 0.168 | 23.67 | 0.986 | | | | |

The memory on one processor of Cray computer is sufficient only for the discretization with coarse grid. For larger problems we report the parallel efficiency related to the results on 2 and 6 processors respectively.

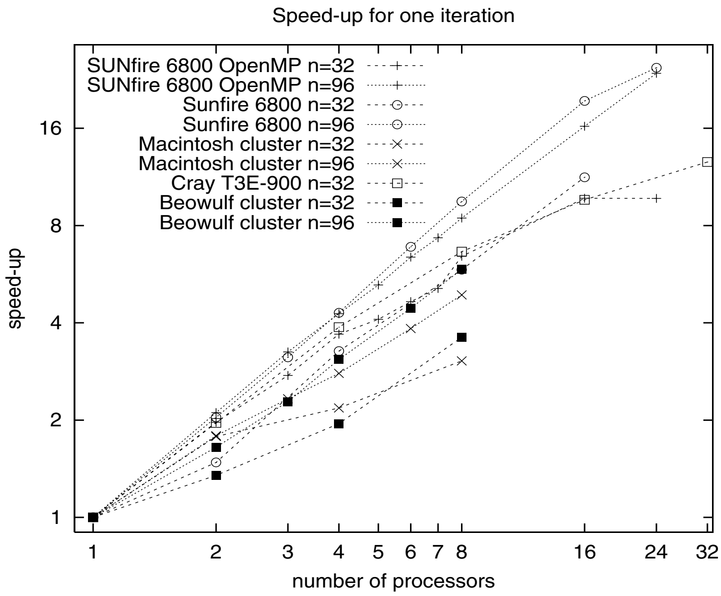


**Fig. 1.** Speed-up for one iteration on three parallel computer systems

Tables 2 and 3 shows results obtained on SUNfire 6800 and on Cray Opteron cluster. As expected, the parallel efficiency increases with the size of the discrete problems. The parallel efficiency for relatively large problems is above 80% which confirms our general expectations. There exist at least two reasons for the reported high efficiency: (a) the network parameters *start-up time* and *time for transferring of single word* are relatively small for the multiprocessor machines; (b) there is also some overlapping between the computations and the communications in the algorithm. Moreover, the super-linear speed-up can be seen in some of the runs. This effect has a relatively simple explanation. When the number of processors increases, the size of data per processor decreases. Thus the stronger *memory locality* increases the role of the cache memories. (The level 2 cache on the SUNfire 6800 is 8 MB.) The obtained speed-up on 16 processors on SUNfire 6800 in some cases is close to 22. In these cases the whole program is fitted in the cache memory and there is no communication between processors and main memory but only between processors.

Finally, we compare results on Cray, SUN, NEC, and Linux clusters. Fig. 1 shows parallel speed-up for execution of one iteration on different parallel systems.

## 4 Summary

We are concerned with the numerical solution of 3D elliptic problems. After discretization, such problems reduce to the solution of linear systems. We use a preconditioner based on a block-circulant approximation of the blocks of the stiffness matrix. We exploit the fast inversion of block-circulant matrices. The computation and the inversion of these circulant block-factorization preconditioners are highly parallelizable on a wide variety of architectures. The developed code provide new effective tool for solving of large-scale problems in realistic time on a coarse-grain parallel computer systems.

## Acknowledgments

This research was supported by the European Commission through grants number HPRI-CT-1999-00026 (the TRACS Programme at EPCC) and number 506079 (HPC-Europa). The experimental results on Cray T3E and SunFire 6800 were obtained during the period of extended access to the EPCC supercomputing facilities after a TRACS research visit in EPCC and the results on NEC and Cray Opteron cluster were obtained during the HPC-Europa research visit at the University of Stuttgart. It was supported also by grant I-1402/2004 from the Bulgarian NSF and by SEE-GRID grant FP6-2002-Infrastructures-2 #002356.

## References

1. R.H. Chan and T.F. Chan, Circulant preconditioners for elliptic problems, *J. Num. Lin. Alg. Appl.* **1** (1992) 77–101.
2. I. Lirkov and S. Margenov, Conjugate gradient method with circulant block-factorization preconditioners for 3D elliptic problems, in: *Proceedings of the Workshop # 3 Copernicus 94-0820 "HIPERGEOS" project meeting, Rožnov pod Radhoštěm* (1996).
3. I. Lirkov and S. Margenov, Parallel complexity of conjugate gradient method with circulant block-factorization preconditioners for 3D elliptic problems, in: O.P. Iliev, M.S. Kaschiev, Bl. Sendov, and P.V. Vassilevski, eds., *Recent Advances in Numerical Methods and Applications* (World Scientific, Singapore, 1999) 455–463.
4. I. Lirkov, S. Margenov, and P.S. Vassilevski, Circulant block-factorization preconditioners for elliptic problems, *Computing* **53** (1) (1994) 59–74.
5. I. Lirkov, S. Margenov, and L. Zikatanov, Circulant block–factorization preconditioning of anisotropic elliptic problems, *Computing* **58** (3) (1997) 245–258.
6. C. Van Loan, *Computational frameworks for the fast Fourier transform* (SIAM, Philadelphia, 1992).
7. Y. Saad and M.H. Schultz, Data communication in parallel architectures, *Parallel Comput.* **11** (1989) 131–150.
8. M. Snir, St. Otto, St. Huss-Lederman, D. Walker, and J. Dongara, *MPI: The Complete Reference.* Scientific and engineering computation series (The MIT Press, Cambridge, Massachusetts, 1997) Second printing.
9. D. Walker and J. Dongara, MPI: a standard Message Passing Interface, *Supercomputer* **63** (1996) 56–68.
10. Web-site for Open MP tools, http://www.openmp.org