Parallel Performance of an MPI Solver for 3D Elasticity Problems

Ivan Lirkov

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

Abstract. The numerical solution of 3D linear elasticity equations is considered. The problem is described by a coupled system of second order elliptic partial differential equations. This system is discretized by trilinear parallelepipedal finite elements.
The Preconditioned Conjugate Gradient iterative method is used for solving of the large-scale linear algebraic systems arising after the Finite Element Method (FEM) discretization of the problem. Displacement decomposition technique is applied at the first step to construct a preconditioner using the decoupled block-diagonal part of the original matrix. Then circulant block-factorization is used for preconditioning of the obtained block-diagonal matrix. Both preconditioning techniques, displacement decomposition and circulant block-factorization, are highly parallelizable.
A parallel algorithm is invented for the proposed preconditioner. The theoretical analysis of the execution time shows that the algorithm is highly efficient for coarse-grain parallel computer systems.
A portable parallel FEM code based on MPI is developed. Numerical tests for real-life engineering problems in computational geomechanics are performed on a number of modern parallel computers: Cray T3E, Sunfire 6800, and Beowulf cluster. The reported speed-up and parallel efficiency well illustrate the parallel features of the proposed method and its implementation.
Keywords: parallel algorithms, PCG method, preconditioner, circulant matrix, elasticity problem
MSC2000: 65F10, 68W10, 74B05, 74B20, 74S05

1 Introduction

This work concerns new efficient parallel algorithms and the related program software for solving the elasticity problem in computational geomechanics. Typical application problems include the simulations of the foundation of engineering constructions (which transfer and distribute the total loading into the bed soil)

I. Dimov et al. (Eds.): NMA 2002, LNCS 2542, pp. 527–535, 2003.

and the multi-layer media with strongly varying material characteristics. Here, the spatial framework of the construction produces a composed stressed-strained state in active interaction zones. A modern design of cost-efficient construction with a sufficient guaranteed reliability requires to determine the parameters of this stressed-strained state.

The application problems are three dimensional nonlinear elasticity problems which are described mathematically by a system of partial differential equations. A finite element (or finite difference) discretization reduces the partial differential equation problem to a system of linear/nonlinear equations. To make a reliable prediction of the construction safety, which is sensitive to soil deformations, a very accurate model and a large system of sparse linear equations is required. In the real-life applications, the system can be very large containing up to several millions of unknowns. Hence, these problems have to be solved by robust and efficient parallel iterative methods on a powerful multiprocessor machine.

Note that the numerical solution of linear systems is fundamental in the elasticity problem. In fact, nonlinear equations generated from the discretization of the nonlinear elasticity problem have to be solved by an iterative procedure, in which a system of linear equations has to be solved in every step of iteration. Solving these linear systems is usually very time-consuming (costing up to 90% of the total solution time). Hence, developing fast algorithms for solving linear equations becomes the most important and fundamental issue. A highly efficient iterative method for solving linear systems significantly speed up the simulation processes of real application problems. An efficient iterative solver should not only have a fast convergence rate but also a high parallel efficiency. Moreover, the resulting program should be efficiently implemented on modern shared-memory, distributed memory, and shared-distributed memory parallel computers.

2 Elasticity Problems

For simplicity, we mainly study the 3D linear elasticity problem based on the following *two basic assumptions*: (1) the displacements are small, and (2) the material properties are isotropic.

The mathematical formulation of the 3D elasticity problem is described as follows. Let $\underline{u} = (u_1, u_2, u_3)^T$ be the displacement vector and $\underline{p}$ the volume force vector. Here T denotes the transpose of a vector or a matrix. Let us denote the matrices D, G and H by

$$D = \begin{pmatrix} \frac{\partial}{\partial x_1} & 0 & 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_3} \\ 0 & \frac{\partial}{\partial x_2} & 0 & \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_3} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} \end{pmatrix}, \qquad G = \begin{pmatrix} \frac{\partial}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_3} \\ \frac{\partial}{2\partial x_2} & \frac{\partial}{2\partial x_1} & 0 \\ 0 & \frac{\partial}{2\partial x_3} & \frac{\partial}{2\partial x_2} \\ \frac{\partial}{2\partial x_3} & 0 & \frac{\partial}{2\partial x_1} \end{pmatrix},$$

$$H = \begin{pmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu) \end{pmatrix}.$$

Then the strain vector $\underline{\epsilon} = (\epsilon_{11}, \epsilon_{22}, \epsilon_{33}, \epsilon_{12}, \epsilon_{23}, \epsilon_{31})^T$ and the stress vector $\underline{\sigma} = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{31})^T$ are determined by

$$\underline{\epsilon} = G\underline{u}, \quad \text{and} \quad \underline{\sigma} = E^* H \underline{\epsilon}, \tag{1}$$

where $E^* = \frac{E}{(1+\nu)(1-2\nu)}$. Here ν and E are respectively the Poisson ratio and the deformation module.

With the above notation, the 3D elasticity problem on a computational domain Ω, can be described by a coupled system of three differential equations, which is written in the form

$$\begin{cases} D\underline{\sigma} = -\underline{p} & \text{in } \Omega \\ \underline{u} = \underline{u_D} & \text{on } \Gamma_D \\ \sum_{i=1}^{3} \sigma_{ij} n_i = \sigma_{Nj} & \text{on } \Gamma_N, \quad j = 1, 2, 3, \end{cases}$$

where Γ_D and Γ_N are the parts of the boundary of Ω with respectively Dirichlet and Neumann boundary conditions; and $\underline{u_D}$ and $\underline{\sigma_N}$ are respectively the given displacement and stress vectors on the boundaries Γ_D and Γ_N. Here we set $\sigma_{ji} = \sigma_{ij}$ for $i < j$.

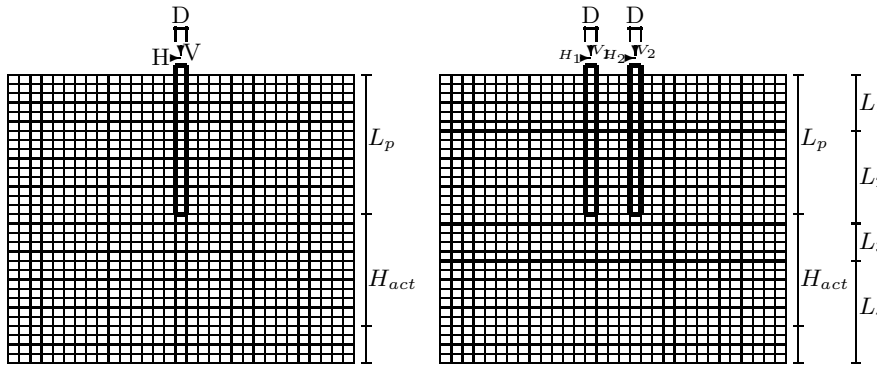
If the Poisson ratio and the deformation module are nonlinear functions, the relations (1) represent the nonlinear nature of the generalized Hooke's law. Here the generalized Hooke's law is specified by the following additional assumption: the Poisson ratio $\nu \in (0, \frac{1}{2})$ is a constant for a given material (soil layer or constructive element). Obviously, this means that the coefficients in the boundary value problem (2) are piece-wise continuous with jumps through the inner boundaries between the different soil layers as well as between the soil and the construction elements.

With a linearization, the nonlinear equations given in (2) can be simplified to a system of three linear differential equations, which is often referred to as the Lamé equations.

Denote Sobolev spaces $\left[H_E^1(\Omega)\right]^3 = \left\{\underline{v} \in \left[H^1(\Omega)\right]^3 : \underline{v}|_{\Gamma_D} = \underline{u_D}\right\}$ and $\left[H_0^1(\Omega)\right]^3 = \left\{\underline{v} \in \left[H^1(\Omega)\right]^3 : \underline{v}|_{\Gamma_D} = 0\right\}$. The variational formulation of the Lamé equations is given below:

$$\text{find } \underline{u} \in \left[H_E^1(\Omega)\right]^3 \text{ such that} \qquad \forall \underline{v} \in \left[H_0^1(\Omega)\right]^3$$

$$\int_\Omega \left[\lambda \, div\, \underline{u} \, div\, \underline{v} + 2\mu \sum_{i=1}^{3} \sum_{j=1}^{3} \epsilon_{ij}(\underline{u}) \, \epsilon_{ij}(\underline{v}) \right] d\Omega = -\int_\Omega \underline{p}^T \underline{v} \, d\Omega + \int_{\Gamma_N} \underline{\sigma_N}^T \underline{v} \, d\Gamma,$$



(a) Problem 1; Cross section of the computational domain Ω. $E_{soil} = 10MPa$, $\nu_{soil} = 0.3$, $E_{pile} = 31500MPa$, $\nu_{pile} = 0.2$

(b) Problem 2; Cross section of the computational domain Ω. $E_{L_1} = 5.2MPa$, $\nu_{L_1} = 0.4$, $E_{L_2} = 9.4MPa$, $\nu_{L_2} = 0.35$, $E_{L_3} = 14.0MPa$, $\nu_{L_3} = 0.25$, $E_{L_4} = 21.4MPa$, $\nu_{L_4} = 0.2$.

Fig. 1. Benchmark problems

where $\lambda > 0$ and $\mu > 0$ are the Lamé coefficients. Here $div\, \underline{u}$ is the divergence of the vector $\underline{u}$. The relations between the elasticity modulus E, ν and the material parameters λ, μ are $\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$. We restrict our considerations to the case $\Omega = [0, x_1^{max}] \times [0, x_2^{max}] \times [0, x_3^{max}]$, where the boundary conditions on each of the sides of Ω are of a fixed type. The benchmark problems from [2] are used in the reported numerical tests. These benchmarks represent the model of a single pile in a homogeneous sandy clay soil layer (see Fig. 1(a)) and two piles in a multi-layer soil media (Fig. 1(b)). An uniform grid is used with n_1, n_2 and n_3 grid points along the coordinate directions. Then the stiffness matrix K can be written in a 3×3 block form where the blocks K_{ij} are sparse block–tridiagonal matrices of a size $n_1 n_2 n_3$.

3 DD CBF Preconditioning

The preconditioning technique used in this work is described in details in [4,3]. The theoretical analysis of the execution time of the proposed parallel algorithm is published in [4]. Here we will only sketch the construction of the preconditioner.

First, we use the approach known as *displacement decomposition* (see, e.g., [1]) to define the preconditioner M_{DD} of the matrix K. We introduce the auxiliary Laplace equation $-u_{x_1x_1} - u_{x_2x_2} - u_{x_3x_3} = f$, with boundary conditions corresponding to the considered coupled elasticity problem. This Laplace equation is discretized by the same brick finite elements as the original problem, and

K_0 is the obtained stiffness matrix. Then $M_{DD} = diag(K_0,\ K_0,\ K_0)$. The next step in our construction is to substitute in M_{DD}, K_0 by A_0, where A_0 stands for the Laplace stiffness matrix corresponding to linear finite elements. Now, let us denote by M_0 the Circulant Block-Factorization (CBF) preconditioner (see [5,4,3]) for A_0. At the last step of our construction we substitute in M_{DD}, K_0 by M_0, and get the DD CBF preconditioner defined by:

$$M_{DD\ CBF} = diag(M_0,\ M_0,\ M_0).$$

The following estimate of the condition number of the preconditioned matrix is derived in [3]

$$\kappa(M_{DD\ CBF}^{-1} K) = \mathcal{O}\left(\frac{n_{\max}}{1 - 2\nu_{\max}}\right)$$

where $n_{\max} = \max(n_1, n_2, n_3)$ and $\nu_{\max} = \max_{\Omega} \nu$.

Remark 1. We have observed in the performed numerical tests that a diagonal scaling of K improves the convergence rate of the iterative method in the case of problems with jumping coefficients.

4 Parallel Tests of the DD CBF Preconditioning FEM Code

In this section we report the results of the experiments executed on three parallel systems. We report here the number of iterations N_{it}, the elapsed time T_p on p processors, the speed-up $S_p = T_1/T_p$, and the parallel efficiency $E_p = S_p/p$. We have used discretizations with $n_1 = n_2 = n_3 = n$ where $n = 32, 48, 64$, and 96. The sizes of the discrete problems are $3n^3$.

The developed parallel code has been implemented in C and the parallelization has been facilitated using the MPI [6,7] and OpenMP 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.

In Table 1 we present results of experiments executed on Cray T3E-900 consisting of 336 Digital Alpha 450 MHz processors, with 64 or 128 MB memory on processor. The memory on one processor of Cray computer is sufficient only for the discretization with $32 \times 32 \times 32$ grid points. For larger problems we report the parallel efficiency related to the results on 6, 16, and 24 processors respectively.

Table 2 shows the results obtained on Beowulf cluster consisting of 17 PC with AMD Athlon processors, 650 MHz, 128 MB memory per computer. The memory per processor is the same and the same approach is used to compute the relative parallel efficiency.

Tables 3 and 4 shows results obtained on a Sunfire 6800 consisting of 24 UltraSPARC-III 750 MHz processors and 48 GB main memory. As expected, the parallel efficiency increases with the size of the discrete problems. The parallel

Table 1. Parallel time (in seconds), speed-up and parallel efficiency on Cray T3E-900.

		Problem 1				Problem 2			
n	p	N_{it}	T_p	S_p	E_p	N_{it}	T_p	S_p	E_p
32	1	112	124.242			471	479.214		
	2		61.371	2.02	1.012		236.371	2.03	1.014
	4		31.029	4.00	1.001		119.153	4.02	1.005
	8		15.973	7.78	0.972		60.529	7.92	0.990
	16		8.741	14.21	0.888		32.795	14.61	0.913
	32		5.104	24.34	0.761		18.991	25.23	0.789
48	6	607	462.692			1118	845.019		
	8		344.337		1.008		626.818		1.011
	12		232.377		0.996		424.325		0.996
	16		176.318		0.984		300.292		1.055
	24		121.932		0.949		206.756		1.022
	48		65.336		0.885		118.534		0.891
64	16	771	491.822			1253	793.226		
	32		245.944		1.000		405.173		0.979
	64		128.406		0.958		209.953		0.945
96	24	1164	1921.650			1941	3190.150		
	32		1424.960		1.011		2361.410		1.013
	48		960.577		1.000		1592.680		1.002

Table 2. Parallel time (in seconds), speed-up and parallel efficiency on Beowulf cluster.

		Problem 1			Problem 2		
n	p	T_p	S_p	E_p	T_p	S_p	E_p
32	1	120.962			486.297		
	2	64.125	1.89	0.943	257.513	1.89	0.944
	4	36.656	3.30	0.825	146.807	3.31	0.828
	8	18.783	6.44	0.805	74.887	6.49	0.812
	16	18.446	6.56	0.410	73.772	6.59	0.412
48	3	1006.950			1721.350		
	4	770.222		0.981	1409.420		0.916
	6	530.166		0.950	902.080		0.954
	8	537.809		0.702	984.016		0.656
	12	517.499		0.486	936.786		0.460
	16	395.148		0.478	674.949		0.478
64	8	1147.780			1909.620		
	16	1385.660		0.414	2304.120		0.414

Table 3. Parallel time (in seconds), speed-up and parallel efficiency on Sunfire 6800 using OpenMP.

		Problem 1			Problem 2		
n	p	T_p	S_p	E_p	T_p	S_p	E_p
32	1	68.271			265.898		
	2	35.388	1.93	0.965	138.818	1.92	0.958
	3	24.884	2.74	0.915	99.436	2.67	0.891
	4	18.527	3.68	0.921	71.172	3.74	0.934
	5	15.423	4.43	0.885	58.423	4.55	0.910
	6	12.594	5.42	0.904	48.040	5.53	0.922
	7	9.627	7.09	1.013	36.469	7.29	1.042
	8	6.802	10.04	1.255	25.033	10.62	1.328
	16	4.013	17.01	1.063	14.519	18.31	1.145
	24	4.401	15.51	0.646	17.889	14.86	0.619
48	1	1601.360			2726.980		
	2	803.957	1.99	0.996	1476.360	1.85	0.924
	3	551.950	2.90	0.967	1000.590	2.73	0.908
	4	407.601	3.93	0.982	739.007	3.69	0.923
	5	347.410	4.61	0.922	635.786	4.29	0.858
	6	277.599	5.77	0.961	479.173	5.69	0.949
	7	246.424	6.50	0.928	414.200	6.58	0.941
	8	208.451	7.68	0.960	376.430	7.24	0.906
	16	108.478	14.76	0.923	184.147	14.81	0.926
	24	78.358	20.44	0.852	136.314	20.01	0.834
64	1	4264.670			7087.930		
	2	2174.790	1.96	0.980	3578.770	1.98	0.990
	3	1479.230	2.88	0.961	2396.550	2.96	0.986
	4	1073.160	3.97	0.993	1751.840	4.05	1.011
	5	902.729	4.72	0.945	1455.900	4.87	0.974
	6	758.351	5.62	0.937	1219.490	5.81	0.969
	7	679.034	6.28	0.897	1111.130	6.38	0.911
	8	551.936	7.73	0.966	894.441	7.92	0.991
	16	294.812	14.47	0.904	489.250	14.49	0.905
	24	269.942	15.80	0.658	433.576	16.35	0.681
96	1	27668.500			45917.300		
	2	13881.100	1.99	0.997	23092.600	1.99	0.994
	3	9374.630	2.95	0.984	15543.400	2.95	0.985
	4	6909.890	4.00	1.001	11521.600	3.99	0.996
	5	5803.100	4.77	0.954	9647.530	4.76	0.952
	6	4740.550	5.84	0.973	7825.420	5.87	0.978
	7	4142.120	6.68	0.954	6808.080	6.74	0.964
	8	3471.910	7.97	0.996	5799.240	7.92	0.990
	16	1825.810	15.15	0.947	3017.930	15.21	0.951
	24	1504.800	18.39	0.766	2461.980	18.65	0.777

Table 4. Parallel time (in seconds), speed-up and parallel efficiency on Sunfire 6800 using MPI.

		Problem 1			Problem 2		
n	p	T_p	S_p	E_p	T_p	S_p	E_p
32	1	78.98			313.33		
	2	43.68	1.81	0.904	168.02	1.86	0.932
	4	20.21	3.91	0.977	79.13	3.96	0.990
	8	8.10	9.75	1.219	26.72	11.73	1.466
	16	3.71	21.28	1.330	12.93	24.24	1.515
48	1	1718.96			2973.82		
	2	864.19	1.99	0.995	1502.07	1.98	0.990
	3	582.46	2.95	0.984	990.31	3.00	1.001
	4	436.42	3.94	0.985	747.80	3.98	0.994
	6	291.81	5.89	0.982	534.42	5.56	0.927
	8	212.68	8.08	1.010	386.37	7.70	0.962
	16	109.80	15.65	0.978	201.43	14.76	0.923
	24	72.59	23.68	0.987	132.34	22.47	0.936
64	1	4847.60			8075.46		
	2	2351.42	2.06	1.031	3869.54	2.09	1.043
	4	1116.15	4.34	1.086	1830.77	4.41	1.103
	8	560.48	8.65	1.081	889.14	9.08	1.135
	16	284.02	17.07	1.067	465.98	17.33	1.083
96	1	30949.00			51284.61		
	2	15197.60	2.04	1.018	25321.60	2.03	1.013
	3	10145.80	3.05	1.017	16848.80	3.04	1.015
	4	7493.50	4.13	1.033	12545.60	4.09	1.022
	6	5006.49	6.18	1.030	8367.84	6.13	1.021
	8	4029.95	7.68	0.960	6644.44	7.72	0.965
	16	1862.06	16.62	1.039	3102.48	16.53	1.033
	24	1412.54	21.91	0.913	2350.58	21.82	0.909

efficiency is above 90% (except the cases where the number of processors do not divide the size of the discrete problem) 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 Mbyte.)

Finally, we compare results on Cray, Sunfire, and Beowulf cluster with our previous results (see [3]) on SGI Origin 2000, SUN Ultra-Enterprise, PowerPC and Alpha clusters. Fig. 2 shows parallel speed-up for execution of one iteration on different parallel systems.

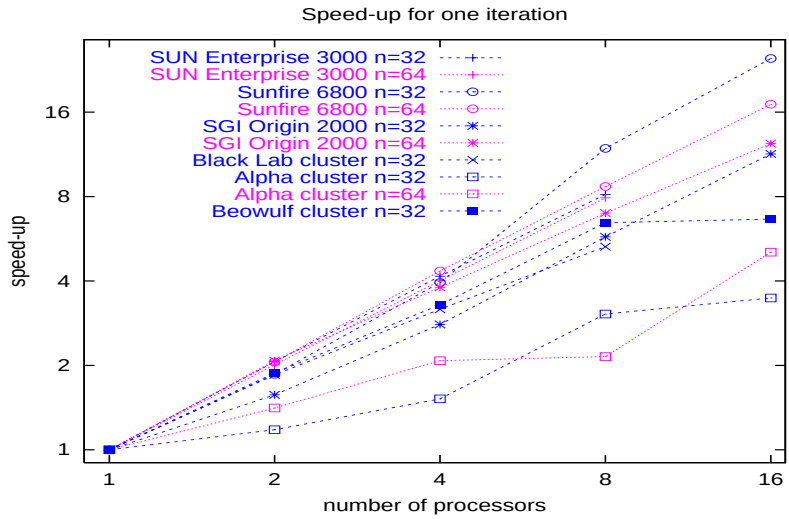


Fig. 2. Speed-up for one iteration

Acknowledgments

This research was supported by the European Commission through grant number HPRI-CT-1999-00026 (the TRACS Programme at EPCC)". The development of the code started while the author was on a TRACS research visit in EPCC, and was completed during the period of extended access to the EPCC supercomputing facilities. It was supported also by grant I-1001/2000 from the Bulgarian NSF and by Center of Excellence BIS-21 grant ICA1-2000-70016.

References

1. Blaheta, R.: Displacement decomposition-incomplete factorization preconditioning techniques for linear elasticity problems. Num. Lin. Alg. Appl., 1 (1994) 107–128.
2. Georgiev, A., Baltov, A., Margenov, S.: Hipergeos benchmark problems related to bridge engineering applications. PROJECT REPORT HG CP 94–0820–MOST–4.
3. Lirkov, I.: MPI solver for 3D elasticity problems. Mathematics and Computers in Simulation, 60, 3-6 (2003) 509–516.
4. Lirkov, I., Margenov, S.: MPI parallel implementation of CBF preconditioning for 3D elasticity problems. Mathematics and Computers in Simulation, 50 (1999) 247–254.
5. Lirkov, I., Margenov, S., Vassilevski, P.S.: Circulant block-factorization preconditioners for elliptic problems. Computing, 53 1 (1994) 59–74.
6. Snir, M., Otto, St., Huss-Lederman, St., Walker, D., Dongara, J.: MPI: The Complete Reference. Scientific and engineering computation series, The MIT Press, Cambridge, Massachusetts (1997) Second printing.
7. Walker, D., Dongara, J.: MPI: a standard Message Passing Interface. Supercomputer, 63 (1996) 56–68.