Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

Massively parallel computing and the boundary element method

A.J. Davies
Division of Mathematics, University of Hertfordshire, Hatfield, UK

ABSTRACT

Massively parallel computing sytems belong to the classification *SIMD*, Single Instruction Multiple Data [1]. In such systems a very large number of relatively unsophisticated processors are connected together in some fashion and there is very rapid transfer of data between adjacent processors. Each processor receives the same instruction but works on its own data. Each processor performs a relatively simple task and for this reason the *SIMD* machine has a fine-grained parallelism.

The boundary element method comprises three distinct phases: (i) equation set-up, (ii) equation solution, (iii) recovery of the field variables. All three phases exhibit a fine-grained parallelism [2], [3], [4], [5] and this parallelism has a straightforward mapping onto an *SIMD* architecture

In this paper we shall consider the mapping of integral formulations of boundary-value problems to massively parallel architectures. In particular we shall consider the mapping of the boundary element method to an array processor such as the AMT DAP or the Connection Machine.

PARALLEL COMPUTERS

In the early 1970's the terminology *supercomputer* was introduced to describe those machines whose performance was a significant improvement on that of convential computers. Supercomputers were developed reflecting the two ways in which performance can be increased; either by direct improvements in hardware technology or by parallel calculation. Parallel computer systems comprise a set of sequential processors interlinked in some manner. The way in which a particular parallel computer works is critically dependent on the architecture *i.e.* on the configuration of the processors which comprise the machine.

Flynn's taxonomy for parallel architectures

Flynn [1] classified such computer architectures into one of four classes:

(i) *Single Instruction Single Data (SISD)*
This is the conventional von Neumann, sequential, architecture. A single processor operates on a single item of data.

(ii) *Single Instruction Multiple Data (SIMD)*
Many processors work simultaneously with the same instruction on different data. Such architectures usually comprise large numbers of relatively unsophisticated processors Each processor has its own memory in which each item of the multiple data resides. Most *SIMD* machines comprise very large numbers of processors, usually many thousands, and as such are often called *massively parallel* machines. The connectivity between processors depends on the actual machine but it is usually very tight so that there is rapid interchange of data between neighbouring processors.

(iii) *Multiple Instruction Single Data (MISD)*
In this type of architecture a multiplicity of different instructions is effected on a single item of data.

(iv) *Multiple Instruction Multiple Data (MIMD)*
A variety of processors work independently with different instructions on different data. Such architectures usually comprise a small number of relatively sophisticated processors.

The *grain size* associated with a parallel machine is a measure of the number and the complexity of the basic operations performed on each processor. *SIMD* machines are often classified as *fine-grained* and *MIMD* machines are often classified as *coarse-grained*. All problems can be divided into subproblems and the size and complexity of the subproblems defines the grain size of the problem. The parallel implementation of any particular problem requires that a suitable mapping is found from the problem onto the computer architecture. It is often the case that the parallelism in a problem is not easily identified with that of the computer and it may well require a considerable amount of ingenuity on behalf of the user to exploit it.

In some circumstances, however, the parallelism inherent in the problem is easily identified with that of the computer architecture and this is particularly true of integral formulations. The boundary element method can be considered as either a fine-grained or a coarse-grained problem and both types of parallelism have been implemented. For further details see Davies [6].

SIMD architecture - the DAP

We describe briefly the architecture of a distributed array processor, the DAP. This type of machine exhibits features which are typical of the *SIMD* class of architectures. Other *SIMD* architectures which have been used in boundary element

implementations are the CM2 Connection Machine [7] and the Intel Hypercube [8].

The DAP comprises 4096 simple one-bit processors arranged in a 64×64 array. Each processor has $16k$ bits of store giving a total of $8Mbytes$. The DAP may be programmed in an array processing version of FORTRAN. There is a substantial subroutine library [9], whose style and standard is based on that of the NAG library, which contains routines written especially to make use of the the DAP architecture. The DAP has been used for a variety of boundary element applications [6].

The Connection Machine, CM-2, has 65,536 single bit processors arranged in groups of 16 per chip. These 16-bit processors are connected in a hypercube network and the special *NEWS* facility allows configuration as an n–dimensional array with $n = 0,1,...,31$. In a similar manner to that of the DAP, the Connection Machine has an extensive subroutine library [10]. A good description of the CM-2 architecture as it applies to boundary element analysis is given by Kumar *et al* [7].

INTEGRAL FORMULATIONS OF BOUNDARY-VALUE PROBLEMS

A boundary-value problem defined by a partial differential equation in a domain, V, together with suitable conditions on the closed boundary, S of V, may be written in the form

$$L\phi = f \quad \text{in } V \tag{1}$$

subject to

$$B\phi = g \quad \text{on } S \tag{2}$$

and such boundary-value problems can be written as integral equations.

Poisson-type problems

The generalised Poisson equation may be written in the form

$$\mathrm{div}\left(k(\mathbf{r})\mathrm{grad}\phi(\mathbf{r})\right) = -f(\mathbf{r}) \tag{3}$$

and this equation may be recast as an integral equation using Green's theorem together with a suitable limiting process. The integral equation is

$$c(\mathbf{r})k(\mathbf{r})\phi(\mathbf{r}) = \int_V \left\{\phi(\mathbf{r}')\nabla' k(\mathbf{r}')\cdot\nabla'\left(G(R')\right) + G(R')f(\mathbf{r}')\right\}dV' + \int_S k(\mathbf{r}')\left\{G(R')\frac{\partial}{\partial n'}\left(\phi(\mathbf{r}')\right) - \phi(\mathbf{r}')\frac{\partial}{\partial n'}\left(G(R')\right)\right\}dS' \tag{4}$$

where $\mathbf{R}' = \mathbf{r} - \mathbf{r}'$, the position of the field point, $\mathbf{r}$, relative to the source point, $\mathbf{r}'$. $\hat{\mathbf{n}}'$ is the unit outward normal at the point $\mathbf{r}'$ on the surface S and $G(R')$ is a fundamental solution of the homogeneous form of equation (3).

Other integral formulations

Elastostatic and electromagnetic problems have a variety of integral formulations of which the following have been implemented in a massively parallel environment:

Elastostatics Betti's reciprocal theorem and the Somigliana identity yield the boundary integral equation

$$c_{ij}(\mathbf{r})u_j(\mathbf{r}) = 2\pi\oint_C \left\{U_{ij}(\mathbf{R}')p_j(\mathbf{r}') - P_{ij}(\mathbf{R}')u_j(\mathbf{r}')\right\}ds' \tag{5}$$

for the boundary displacements, $u_j(\mathbf{r})$, and tractions, $p_j(\mathbf{r})$. This equation has been considered by Song *et al* [11] and implemented on the AMT DAP.

Electromagnetics Maxwell's equations and Green's theorem yield the boundary integral equation

$$\left[1 + c(\mathbf{r})\chi\right]\psi(\mathbf{r}) = \frac{\chi}{2\pi}\oint_C \psi(\mathbf{r}')\frac{\partial}{\partial n'}(\ln R')ds' + \psi_s(\mathbf{r}) \tag{6}$$

for the total magnetic scalar potential ψ, where χ is the material susceptibility and ψ_s is the potential associated with an applied source field. This formulation has been considered by Davies [12] and implemented on the ICL DAP.

Potential problems

If $k(\mathbf{r}')$ is constant and $f(\mathbf{r}) \equiv 0$ in V then the partial differential equation is Laplace's equation and the corresponding integral equation is defined only over the boundary, S. We shall illustrate the method by considering two-dimensional potential problems defined over the plane region D bounded by the closed curve C. It is convenient to write the flux variable as q, *i.e.* $\dfrac{\partial \phi}{\partial n} \equiv q$.

Suppose that we have a Dirichlet boundary condition on a section C_0 and a Neumann boundary condition on a section C_1, where $C = C_0 + C_1$. Then the boundary integral equation (4) is given by

$$2\pi c\phi = \oint_C \left\{ \phi \frac{\partial}{\partial n'}(\ln R') - q \ln R' \right\} ds' \tag{7}$$

where we have suppressed the arguments in the functions ϕ, q and C. Internal potentials are given by equation (4), with $c = 1$, as

$$\phi = \frac{1}{2\pi} \oint_C \left\{ \phi \frac{\partial}{\partial n'}(\ln R') - q \ln R' \right\} ds' \tag{8}$$

BOUNDARY ELEMENT FORMULATION

In this section we give a brief description of the formulation of the boundary element equations since the inherent parallelism becomes transparant during the development. We consider the potential problem as described in the previous section because this

simple problem exhibits the inherent parallelism which is typical of the boundary element method.

Choose a set, $S = \{1,2,...,N\}$, of *collocation points* on the boundary, see Figure 1, at which we seek the values, ϕ_i and q_i, of the potential and the flux. Associated with each collocation point is a suitable basis function so that we have the set $\{u_j(s): j = 1,2,...,N\}$ of linearly independent functions defined in terms of the distance, s, around the boundary.

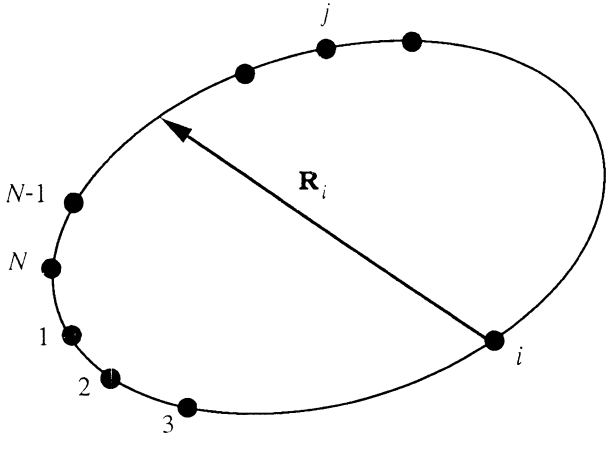

Figure 1 Boundary collocation points

We now consider the following approximations to ϕ and q on the boundary:

$$\tilde{\phi}(s) = \sum_{j=1}^{N} u_j(s)\phi_j \quad \text{and} \quad \tilde{q}(s) = \sum_{j=1}^{N} u_j(s)q_j. \tag{9}$$

Substitute the expressions for $\tilde{\phi}$ and $\tilde{q}$ in equation (7) and choose the boundary field point, $\mathbf{r}$, to be successively each of the N nodes to obtain:

$$2\pi c_i \phi_i \approx \oint_C \left\{ \sum_{j=1}^{N} u_j(s')\phi_j \frac{\partial}{\partial n'}(\ln R_i) - \sum_{j=1}^{N} u_j(s') q_j \ln R_i \right\} ds', \quad (i = 1,2,\ldots,N) \tag{10}$$

which we write as

$$\sum_{j=1}^{N} H_{ij}\phi_j + \sum_{j=1}^{N} G_{ij} q_j = 0, \qquad (i = 1,2,\ldots,N) \tag{11}$$

where $$H_{ij} = \oint_C u_j(s') \frac{\partial}{\partial n'}(\ln R_i) ds' - 2\pi c_i \delta_{ij} \tag{12}$$

and $$G_{ij} = -\oint_C u_j(s') \ln R_i ds'$$

If we write the set of equations (11) in matrix form as

$$\mathbf{H}\boldsymbol{\phi} + \mathbf{G}\mathbf{q} = \mathbf{0} \tag{13}$$

then we partition the matrices in equation (13) according to the type of boundary condition which occurs at the node i, i.e. we write

$$\begin{bmatrix} \mathbf{H}^{(0)} & \mathbf{H}^{(1)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^{(0)} \\ \boldsymbol{\phi}^{(1)} \end{bmatrix} + \begin{bmatrix} \mathbf{G}^{(0)} & \mathbf{G}^{(1)} \end{bmatrix} \begin{bmatrix} \mathbf{q}^{(0)} \\ \mathbf{q}^{(1)} \end{bmatrix} = \mathbf{0} \tag{14}$$

where the superscripts 0 and 1 refer to Dirichlet and Neumann conditions respectively. Since $\boldsymbol{\phi}^{(0)}$ and $\mathbf{q}^{(1)}$ are known we re-arrange equations (14) as

$$\begin{bmatrix} \mathbf{G}^{(0)} & \mathbf{H}^{(1)} \end{bmatrix} \begin{bmatrix} \mathbf{q}^{(0)} \\ \boldsymbol{\phi}^{(1)} \end{bmatrix} = -\begin{bmatrix} \mathbf{H}^{(0)} & \mathbf{G}^{(1)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^{(0)} \\ \mathbf{q}^{(1)} \end{bmatrix}$$

i.e. we have a system of equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{15}$$

We note here that $\mathbf{A}$ is a fully populated matrix which, in general, is not symmetric.

The solution of the system of equations (15) leads to pairs of values (ϕ_i, q_i) at node i, $(i=1,2,...,N)$ from which we can obtain the interior potentials using equation (8)

$$\phi_k \approx \frac{1}{2\pi}\oint_C \left\{ \tilde{\phi}\frac{\partial}{\partial n'}(\ln R_k) - \tilde{q}\ln R_k \right\} ds'$$

where $\mathbf{R}_k$ is the position vector of the boundary point $\mathbf{r}'$ relative to the interior field point $\mathbf{r}_k$.

Finally, using equation (9) for $\tilde{\phi}$ and $\tilde{q}$ we obtain

$$\phi_k \approx \frac{1}{2\pi}\sum_{j=1}^{N}\left\{\oint_C u_j(s')\frac{\partial}{\partial n'}(\ln R_k)ds'\right\}q_j - \frac{1}{2\pi}\sum_{j=1}^{N}\left\{\oint_C u_j(s')\ln R_k ds'\right\}q_j \qquad (16)$$

Basis functions

We choose the basis functions in a piecewise manner. Boundary element implementations in massively parallel environments have included piecewise constant [13], linear [2] and quadratic [3] formulations. We shall consider here a piecewise linear formulation in which the basis functions are the usual *hat* functions.

The terms in the system matrices are of the form

$$H_{ij} = \hat{H}_{ij} - 2\pi c_i \delta_{ij}$$

$$\text{with } \hat{H}_{ij} = \left\{\int_{C_{j-1}} + \int_{C_j}\right\} u_j(s')\frac{\partial}{\partial n'}(R_j)ds'$$

$$= h_{ij}^{(1)} + h_{ij}^{(2)} \text{ say} \qquad (17)$$

and

$$G_{ij} = \left\{\int_{C_{j-1}} + \int_{C_j}\right\} u_j(s')\ln R_j ds'$$

$$= g_{ij}^{(1)} + g_{ij}^{(2)} \text{ say} \qquad (18)$$

If d_{ij} is the distance from the base node i onto the target element j then, using an N_g Gauss quadrature for the integrals, we have

$$\hat{H}_{ij} \approx d_{ij-1}\left(\sum_{g=1}^{N_g} u_j(s_g)\frac{1}{\left(R_{ij-1}^2\right)_g}\omega_g\right)\frac{l_{j-1}}{2} + d_{ij}\left(\sum_{g=1}^{N_g} u_j(s_g)\frac{1}{\left(R_{ij}^2\right)_g}\omega_g\right)\frac{l_j}{2}$$

$$G_{ij} \approx -\left(\sum_{g=1}^{N_g} u_j(s_g)\ln\left(R_{ij-1}\right)_g\omega_g\right)\frac{l_{j-1}}{2} - \left(\sum_{g=1}^{N_g} u_j(s_g)\ln\left(R_{ij}\right)_g\omega_g\right)\frac{l_j}{2} \qquad (19)$$

Recovery of the internal potentials

Once the nodal values of potential and flux have been determined we can obtain values of the internal potentials using equation (16) with an N_g Gauss quadrature

$$\phi_k \approx \frac{1}{2\pi}\sum_{j=1}^{N}\left[\left(\overline{H}_{kj} + \overline{H}_{kj+1}\right)\phi_j + \left(\overline{G}_{kj} + \overline{G}_{kj+1}\right)q_j\right]$$

where

$$\overline{H}_{kj} \approx d_{kj}\left(\sum_{g=1}^{N_g} u_j(s_g)\frac{1}{\left(R_{kj}^2\right)_g}\omega_g\right)\frac{l_j}{2}$$

$$\overline{G}_{kj} \approx -\left(\sum_{g=1}^{N_g} u_j(s_g)\ln\left(R_{kj}\right)_g\omega_g\right)\frac{l_j}{2} \qquad (20)$$

Overview of the boundary element method

There are essentially three phases in the boundary element method:

(i) The *set-up* phase in which the coefficients H_{ij}, G_{ij}, $\overline{H}_{kj}$ and $\overline{G}_{kj}$ are evaluated.

(ii) The *solution* of the system of algebraic equations.

(iii) The *recovery* of the internal potentials.

All three phases exhibit a parallelism which may be mapped onto a massively parallel architecture.

PARALLELISATION OF THE METHOD

A suitable mapping to an array processor involves a direct correspondence between the system of equations and the array. The coefficient a_{ij} in matrix **A**, equation (15), is associated with the processing element in the (i,j) position and the system is mapped to a single plane.

Typical features of the parallelism are displayed in the calculation of $h_{ij}^{(2)}$ using equation (17), *i.e.*

$$h_{ij}^{(2)} \approx d_{ij}\left(\sum_{g=1}^{N_g} u_j(s_g)\frac{1}{\left(R_{ij}^2\right)_g}\omega_g\right)\frac{l_j}{2}.$$

In order to calculate $h_{ij}^{(2)}$, three loops are required: an outer loop over the base nodes, an intermediate loop over the target elements and an internal loop over the Gauss points. On the array processor the two outer loops are effected simultaneously in parallel and the inner loop only is performed sequentially [14]. $\overline{H}_{ij}$ and $\overline{G}_{ij}$, equation (20), are obtained in an identical manner.

Finally, as well as providing an environment which mimics the parallelism inherent in the boundary element method, it is often the case that the programming language provides code which is more transparent than the equivalent sequential code [2].

MASSIVELY PARALLEL IMPLEMENTATIONS

The first parallel implementation of the boundary element method was described by Symm [13], whose development, on the

ICL DAP, was a short feasibility study which indicated that the DAP provides a highly suitable environment for the boundary element method.

Various authors have described parallel computations only for certain aspects of the boundary element method. In the early attempts, most workers concentrated on the linear equation solution phase. Calitz and du Toit [15] use an integrated array processor, attached to a workstation, to effect the solution phase in an axisymmetric electromagnetic problem.

Complete fine-grained implementations, in which all phases exploit the parallelism, are described by Davies [2],[3],[4], who considered a variety of linear and quadratic element implementations of potential problems on the ICL DAP.

Other massively parallel implementations have been developed as follows:

(i) Lai [16] describes a parallel implementation on the DAP of a panel method for flow around an aerofoil.
(ii) An electromagnetic problem, using linear elements, has been considered by Davies [12] on the DAP.
(iii) A parallel implementation of a quadratic element approach to the solution of axisymmetric elasto-static problems is given by Song *et al* [11].
(iv) In large-scale three-dimensional stress analysis, numerical quadrature is a significant computational feature; Kane *et al* [17] describe an implementation on the Connection Machine, CM2.
(v) The Poisson problem can be formulated as a boundary element method via the dual reciprocity technique and a fine-grained implementation of this method is described by Davies [5].

COMPUTATIONAL PERFORMANCE

A variety of potential and Poisson problems has been considered by Davies [2],[3],[5],[12] and comparisons between typical DAP run times and mainframe sequential run times are given in Table 1.

Problem	size	DAP time (*ms*)	Sequential time (*ms*)
Potential			
linear elements	64×64	160[3]	16000[1]
quadratic elements	128×128	700[3]	37000[1]
Poisson DRM			
linear elements	64×64	315[4]	38430[2]
Magnetostatic			
linear elements	64×64	140[3]	4200[2]
	126×126	590[3]	20100[2]

Table 1 cpu times (*ms*) for a variety of DAP implementations and the corresponding sequential times
[1] DEC 1091, [2] VAX 850, [3] ICL DAP,
[4] AMT DAP with coprocessor

Speed ups vary and depend on the problem; typically they are in the range 30 to 100.

The elastostatic problem described by Song *et al* [11] comprised 32 boundary nodes with a run time of about 1*s* on the AMT DAP and about 20*s* on the ICL 2988, a speed up of the order of 20.

Kumar *et al* [7] consider large-scale three-dimensional stress analysis problems and show that the computation time on

the Connection Machine, CM-2, increases linearly with respect to the number of boundary elements, compared with the quadratic increase for sequential and vector processors.

CONCLUDING REMARKS

We have discussed boundary element implementations on massively parallel (*SIMD*) machines and shown that there is an inherent parallelism in the method which is mapped very easily to fine-grained architectures. Unfortunately large massively parallel computers, such as the DAP and the Connection Machine, are expensive and require an initial familiarisation. However, where they are available, they provide a very attractive alternative to the large and extremely expensive vector-pipeline machines for boundary element implementations.

REFERENCES

1 Flynn M. Some computer organizations and their effectiveness. *IEEE Trans Computing*, C-21, 948-960, 1972.

2 Davies AJ. The boundary element method on the ICL DAP. *Parallel Computing*, 8, 348-353, 1988.

3 Davies AJ. Quadratic isoparametric boundary elements on the ICL DAP - in *Boundary Elements X*, ed Brebbia CA, 3, 657-666, Springer-Verlag, 1988.

4 Davies AJ. Mapping the boundary element method to the ICL DAP - in *CONPAR 88*, eds Jesshope CR and Reinartz KD, 230-237, Cambridge University Press, 1989.

5 Davies AJ. A parallel implementation of the dual reciprocity boundary element method - in *Boundary Elements XIV*, eds Brebbia CA, Dominguez J and Paris F, 2, 613-626, Elsevier, 1992.

6 Davies AJ. Parallel boundary element implementations: a survey. *University of Hertfordshire NOC Technical Report* No. 269, 1993.

7 Kumar BLK, Kane JH, Srinivasan AV and Wilson RB. The influence of massively parallel processing in boundary element computations - in *Proceedings of the International Symposium on Boundary Element Methods*, 500-505, Springer-Verlag, 1989.

8 Drake JB and Gray LJ. Parallel implementation of the boundary element method - in Vector and Parallel Computing, eds Dongarra J, Duff I, Gaffney P and McKee S, 88-92, Ellis Horwood, 1989.

9 Liddell HM and Bowgen GSJ. The DAP subroutine library. *Comput. Phys. Commun.*, 26, 311-315, 1982.

10 Thinking Machines. *CM-FORTRAN reference manual*, Thinking Machines Corporation, 1991.

11 Song B, Gay R and Parsons B. Parallel processing of quadratic boundary elements. To appear in *Engng. Anal. with Bdry. Elements*, 1993.

12 Davies AJ. The boundary element method for magnetostatic problems using a distributed array processor - in *Boundary Elements in Mechanical and Electrical Engineering*, eds Brebbia CA and Chaudouet-Miranda A, 407-415, Springer-Verlag, 1990.

13 Symm GT. Boundary elements on a distributed array processsor. *Engng. Anal.,* 1, 162-165, 1984.

14 Davies AJ. Parallelism in the boundary element method; fine grain and coarse grain - in *Applications of Supercomputers II*, eds Brebbia CA, Howard D and Peters A, 61-72, Elsevier, 1991.

15 Calitz MF and du Toit AG. CAD system for cylindrically symmetric electric devices. *IEEE Trans Magnetics*, MAG-24, 427-430, 1988.

16 Lai CM. A parallel panel method for the solution of fluid flow past an aerofoil - in *CONPAR 88*, eds Jesshope CR and Reinartz KD, 711-718, Cambridge University Press, 1989.

17 Kane JH, Kumar BLK, Wilson RB and Srinivasan AV. Data Parallel (SIMD) boundary element stress analysis. Paper presented at *IABEM-92*, University of Colorado, US, 1992.