A PARALLELIZATION TECHNIQUE FOR PRECONDITIONED BOUNDARY-ELEMENT-METHOD SOLVERS

Y.-J. Yang and H.-K. Tang

Department of Mechanical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract—In this paper, a methodology for parallelizing boundary-element-method (BEM) solvers that employ either the multipole or the FFT algorithms are presented. The basic idea is to parallelize the process of creating the preconditioner that is required to efficiently solving the system iteratively. Two simulated case studies are presented. The comparisons of the performances between three parallelization approaches, which use different models to balance the load, are also presented. The simulated results of a testing sphere structure, whose BEM panels are uniformly distributed, show that the computation time can be effectively reduced by parallel processing without any compromise in accuracy. Furthermore, our proposed approaches also achieve better speed-ups than previous works for a field-emission device whose BEM panels are highly non-uniformly distributed.

1 Introduction

2 Mathematical Background

- 2.1 Problem Formulation Using BEM
- 2.2 Iterative Method and Preconditioner

3 Parallelization of Preconditioner Construction

- 3.1 Preliminary Parallelization Implementation: Without Load Balance
- 3.2 Parallelization with Load Balance: Sorting Based on Submatrix Sizes
- 3.3 Parallelization with Load Balance: Sorting Based on Cost Model

4 Results and Discussion

4.1 Case I: A Translating Sphere

4.2 Case II: A Field Emission Device

5 Conclusion

Acknowledgment

References

1. INTRODUCTION

Electrostatic computations, such as capacitance and electrostatic field/force calculations, are vital for modeling of electron devices and microelectromechanical systems (MEMS). The boundary-element-method (BEM) is widely used to implement capacitance/electrostatic solvers [1, 2] because it avoids meshing the open space that surrounds the simulated structures (conductors/dielectrics). Various BEM electrostatic solvers have been developed during the past years. Most of these solvers employ hierarchical approximation methods such as the multipole or the FFT algorithms [3–7] for constructing sparse-like system matrices, and require adequate pre-conditioning matrices for smoothly solving the system matrices using iterative methods [8, 9].

The computational costs of these approaches become very expensive as the geometries of simulated devices are complicated. One of the best ways to reduce the total computational time is to employ parallelization techniques. There have been many works on parallelizing the FFT or the multipole algorithms for BEM electrostatic solvers. For the FFT approaches, Aluru *et al.* parallelized the matrix-vector product step for the precorrected FFT (P-FFT) algorithm [10] using the IBM SP2. Li *et al.* [11] proposed a parallelization algorithm based on the incident angles of scattering problems, and showed that the performance of this approach is better than parallelizing the P-FFT algorithm.

For the multipole approaches, Wang *et al.* [12] parallelized the fast multipole algorithm on a Transputer network. The parallel calculations are implemented by balancing the workload in every level of cubes and using the pipeline communication mode. Yuan *et al.* [13] proposed a generalized cost function model that can be used to accurately measure the workload of each cube. Also, two adaptive partitioning schemes for balancing the load were presented in the work, and the results on a variety of platforms were also reported.

In this paper, instead of parallelizing the multipole or the FFT algorithm, we focus on parallelizing the process of creating

the preconditioners that will be used for iteratively solving the BEM system matrices. It is well-known that one of the main bottlenecks for many BEM calculations is related to the solutions of the large dense and nonsymmetric system matrices. Therefore, iterative methods (such as GMRES [9]) for solving nonsymmetric linear systems are usually employed. Furthermore, preconditioning techniques are frequently essential for iterative methods to converge within a reasonable number of iterations, especially for the cases that small and large structures co-exist in the same simulated model [7, 14, 15] (e.g., in the same model, the size of the largest BEM panel and the size of the smallest BEM panel differ significantly).

The multipole and the FFT algorithms require the process of partitioning the whole computational domain into many small subdomains when the BEM system matrices are constructed. The performance of creating a preconditioner heavily depends on how the computational domain is partitioned. Typically, if the whole computational domain is divided into a *smaller* number of subdomains (i.e., a lower partitioning depth), the size of the smallest subdomain is *larger*, which in turn results in *larger* sizes of the submatrices which are required to be inverted for creating a preconditioner matrix. Therefore, the overall computational bottleneck frequently occurs at the step of creating the preconditioner [16] for the cases with lower partitioning depths. On the other hand, with a higher partitioning depth, the computational cost for creating the preconditioner usually decreases. However, simulations with higher partitioning depths usually give worse accuracy because some of the BEM panels might be larger than (or comparable to) the size of the smallest subdomains [15].

By using the MPI functionality [17, 18], we parallelize the preconditioned BEM solver Fastlap [4] that employs the multipole technique for solving general three-dimensional Laplace problems. The preconditioner of the Fastlap is computed by inverting a group of submatrices extracted from the BEM system matrix. Three different methods for parallelizing the preconditioner calculations are proposed. We also formulate a cost model that accounts for the costs of submatrix inversion and the network transmission. Two different devices, a testing sphere and a field-emission device, are simulated using a PC cluster installed with the Linux operation system and the MPI library.

2. MATHEMATICAL BACKGROUND

2.1. Problem Formulation Using BEM

In this work, we consider the first-kind integral equation of the potential theory for a single-layer surface charge density that is

generated by the solution of the exterior Dirichlet problem 0 [19]. The surface charge density σ satisfies the integral equation:

$$\psi(\bar{x}) = \int_{surfaces} \sigma(\bar{x}') \frac{1}{4\pi\epsilon_0 \|\bar{x} - \bar{x}'\|} da', \quad \bar{x} \in surfaces. \quad (1)$$

where $\psi(\bar{x})$ is the surface potential, σ is the surface charge density, da' is incremental surface area, $\bar{x}, \bar{x}' \in R^3$, and $\|\bar{x}\|$ is the Euclidean length of x given by $\sqrt{x_1^2 + x_2^2 + x_3^2}$. We can compute an approximation of σ by assuming that a charge q_i on the i -th panel is uniformly distributed at its centroid (the piece-wise constant collocation scheme). Equation (1) can then be formulated as [8]:

$$\bar{\bar{P}}\bar{q} = \bar{p} \quad (2)$$

where $\bar{\bar{P}} \in R^{n \times n}$ is the system matrix, $\bar{q} \times R^n$ is the vector of unknown panel charges, $\bar{p} \in R^n$ is the vector of known panel potentials, and

$$P_{ij} = \frac{1}{a_j} \int_{panel_j} \frac{1}{4\pi\epsilon_0 \|\bar{x}_i - \bar{x}'\|} da' \quad (3)$$

where $\bar{x}_i$ is the center of the i -th panel and a_j is the area of the j -th panel. The unknown charge vector $\bar{q}$ can be obtained by iteratively solving Equation (2), and the charge density σ can be obtained directly from q_i . Because the potential on each panel is affected by the surface charges on all the other panels, the matrix $\bar{\bar{P}}$ is a dense matrix. Given N panels, the cost of forming $\bar{\bar{P}}$ is $O(N^2)$. When an iterative method is used to solve (2), the cost of computing the dense matrix-vector products for each iteration is $O(N^2)$ [8]. The acceleration technique in the Fastlap is to use the multipole method for fast evaluation of each matrix-vector product with cost $O(N)$. The details of the multipole algorithm can be found in [20].

2.2. Iterative Method and Preconditioner

The Fastlap solves Equation (2) using an iterative method called GMRES [9]. This method evaluates the next candidate of solution from all the residuals in previous iterations. The new iterate $\bar{q}^{i+1}$ for each iteration step (except the first one) is:

$$\bar{q}^{i+1} = \bar{q}^0 + \bar{z}^{i+1}, \quad i = 0, 1, \dots, N-2 \quad (4)$$

where $\bar{q}^0$ is the initial guess, and $\bar{z}^{i+1}$ is a vector in the space of the Krylov vectors. The $\bar{P}\bar{q}^k$ in each iteration is accelerated by the fast multipole algorithm [20]. Besides, a preconditioner matrix $\bar{A}$ is used to increase the convergence rate. Equation (2) is multiplied by the preconditioner $\bar{A}$ on both sides:

$$\bar{A}\bar{P}\bar{q} = \bar{A}\bar{p} \quad (5)$$

where the preconditioner matrix $\bar{A}$ is computed by inverting a sequence of submatrices extracted from the $\bar{P}$ matrix, as shown in Fig. 1. Typically, the $\bar{A}$ matrix is close to $\bar{P}^{-1}$ [8]. The size of each submatrix is strongly dependent on the partitioning depth and the panel distribution. Usually the sizes of submatrices become bigger when partitioning depth decreases. The cost of inverting these submatrices by Gauss elimination is $O(N_S^3)$, where N_S is the size of each submatrix. It is clear that the overall computational cost will be detrimentally affected if some of N_S are relatively large (e.g., a few hundreds). On the other hand, as the partitioning depth increases, the efficiency of inverting submatrices increases, while the error of iteratively solving (10) might also increase because some of the BEM Panels will be larger than (or comparable to) the size of the smallest sub-domains. Therefore, if the preconditioning strategy can be parallelized, the computational time can be effectively reduced without compromise in accuracy at the conditions of small partitioning depths.

Figure 1. The schematic of forming the pre-conditioner matrix $\bar{A}$ from system matrix $\bar{P}$.

3. PARALLELIZATION OF PRECONDITIONER CONSTRUCTION

3.1. Preliminary Parallelization Implementation: Without Load Balance

The preconditioner is in fact a collection of the inversions of the submatrices that are originally extracted from the BEM system matrix. Inverting those submatrices is the most expensive step in the process of creating a preconditioner, especially when the partitioning depth is small or the BEM panel distribution is non-uniform. Our first parallelization approach is to distribute those submatrices to different nodes of a PC cluster for inverting, based on the sequence in which those submatrices are extracted from the system matrix (i.e., this sequence is strongly dependent on how the fast-multipole algorithm hierarchically divides the computational domain into smaller cubes). Each node of the PC cluster computes its own received submatrices and sends the results to a server node after the computations are done. Finally, the server node constructs the preconditioner based on those inverted submatrices.

3.2. Parallelization with Load Balance: Sorting Based on Submatrix Sizes

The computing times for inverting the submatrices usually differ significantly since the sizes of the submatrices are different, especially for the cases with non-uniform distribution of panels. As a result, we rearrange the sequence of distributing the submatrices to cluster nodes based on the sizes of the submatrices (sorting). After the rearrangement, the n -th sub-matrix is assigned to the processor n (or the remainder of n divided by N_P , $(n \bmod N_P)$, where N_P is the number of processors). Fig. 2 shows how the performance can be improved after the rearrangement. Consider a system with six submatrices of different sizes. Fig. 2(a) indicates that only one processor (P_0) is used to compute the submatrices (i.e., without parallelization). The computational costs of the submatrices are indicated with different shades. The total required computational cost is 12 units. Fig. 2(b) indicates that if the submatrices are distributed to the two nodes without rearranging their orders, the required computing time is not optimal. Fig. 2(c) shows that if the submatrices are distributed to the nodes after rearrangement (sorting), the performance may reach the maximum (an ideal case). Note that in this work the sequence of distributing submatrices is sorted by using the quick sort algorithm whose computational cost is $O(N_{sub} \log N_{sub})$,

Figure 2. (a) Given six sub-matrices of different sizes (different required computing times). It costs 12 units by one processor. (b) and (c) show the total computing times by two nodes without and with rearranging the distributing sequences.

where N_{sub} is the total number of submatrices. Our study shows that this cost is negligible (less than 0.1% of the total computation time) in the overall computation even for the case when the partition depth is equal to 5.

3.3. Parallelization with Load Balance: Sorting Based on Cost Model

The parallelization based on the submatrix sizes described in previous subsection might not be ideal for many cases. First of all, since the cost of inverting a dense matrix with size N by the Gauss elimination is $O(N^3)$, distributing submatrices based on their matrix sizes, which implies an $O(N)$ cost for inverting, is possibly inadequate. Secondly, this method will easily results in unbalanced load for the cases that BEM panels are *highly non-uniformly* distributed since the sizes of the submatrices differ significantly [22]. Therefore, we propose a cost model to estimate the computation cost of each submatrices. The cost of inverting the i -th submatrix is defined as

$$c(i) = (N_s(i))^3 + K_n \times (N_s(i))^2 \quad (6)$$

where $N_s(i)$ is the size of the i -th submatrix, K_n is the weighting factor. The first term in the equation is the cost for computing the inversion, and the second term is the cost of network communication. Note that K_n is zero when the i -th submatrix is calculated by the server of the PC cluster, because no network communication is required. A larger weighting factor K_n indicates a higher cost of network communication, and vice versa. The typical value of $N_s(i)$ is greater than 100 for a partition depth that is 3 or less. Therefore, the second term of Equation (6) is insignificant unless K_n is on the same order of magnitude of $N_s(i)$. Our preliminary numerical experiments show that the network communication of our PC cluster is efficient enough (i.e., K_n is close to unit) so that the second term of Equation (6) can be neglected. The algorithm with the load balance is briefly summarized in Fig. 3.

4. RESULTS AND DISCUSSION

We parallelize the Fastlap by adding the MPICH functionality into the original source code. Since we only focus on the parallelization of the inversion computation for creating preconditioner, only a mild modification on the source code is required. In this section, we present the simulated results using different parallelization approaches described in previous section. All the simulations were performed on an eight-node PC cluster. Each node has an Intel Xeon 2.8 GHz processor and 2 GB memory. These eight nodes are interconnected through a Gigabit Ethernet network. The first simulated case is a sphere under various boundary conditions [4], and the second case is a field emission device [14, 15].

4.1. Case I: A Translating Sphere

This case deals with a sphere of a unit radius translating in an infinite fluid with a unit velocity, and is the example provided in the package of the Fastlap source code. Fig. 4 shows the (simplified) 3-D BEM model that is discretized by equal subdivision of the polar and azimuthal angles. A Dirichlet boundary condition is uniformly applied on the sphere. The analytical solution is available in closed form. Fig. 5 shows the relationship between the partitioning depths and the maximum errors, for the cases of four different numbers of BEM panels. The definition of the maximum error E is:

$$E = \frac{\max_{1 \leq i \leq n_s} |simu(i) - exact(i)|}{\max_{1 \leq i \leq n_s} |exact(i)|} \quad (7)$$

Algorithm

```
{
Initialization
    Find all the sub-matrices in every finest level cubes and then sorting according
    to the size of every cube
For the specific server
    for i=1 to ns do /*ns is the number of the sub-matrices*/
        Compute c(i) of the sub-matrices and determine its destination according to the
        cost to make the total cost of each node nearly the same.
    for i=1 to ns do /*assign sub-matrices to all other nodes according to the last step*/
        send those sub-matrices that need to be calculated by other nodes.
    inv(sm); /*sm are those sub-matrices that need to be computed by the server itself*/
    for i=1 to ns do
        receive all the sub-matrices from other nodes.
    Form the pre-conditioner from those inversions of sub-matrices
For the other nodes
    for i=1 to ns do
        receive its own sub-matrices from the server
    inv(sm);
    for i=1 to ns do
        send those sub-matrices back to the server
}
```

Figure 3. Parallelization algorithm of pre-conditioner construction by the cost model to balance the load in each processor.

where n_s is the number of BEM panels, $simu(i)$ and $exact(i)$ are the simulated result and the exact solution of the i -th panel, respectively. Note that the maximum error shown in Fig. 5 has been normalized by the maximum absolute value of the exact solution, which is equal to 1 for this case.

As described in previous section, the partitioning depth is an integer that specifies the level to which the computational domain will be hierarchically decomposed. As the partitioning depth equals to 1, there is only one submatrix (i.e., equal to the original $\overline{\overline{P}}$) to be inverted, which indicates that parallelization will not be applicable. Therefore, these curves start with the partitioning depth of 2. As shown in this figure, the maximum errors decrease with the partitioning depth. Apparently, depths 2 or 3 are required for obtaining accurate results (less than 1% error) for these cases. Notice that very large computational times are required for the simulations with depths 2 and 3 (Table 1) because of the inversion calculations of the submatrices. As shown in Table 2, the cost of evaluating a preconditioner is dominant in the total computing time (over 85%) for the cases

Figure 4. A 3-D boundary element mesh plot of the sphere simulated.

Figure 5. Maximum errors vs. partitioning depths for different numbers of panels.

with low partitioning depths ($d = 2$ or 3). Therefore, for these cases, the application of parallelization for accelerating the process of evaluating the preconditioner becomes advantageous. Fig. 6 shows the relationship between the speed-up and the partitioning depth. The speed-up and the efficiency are defined as

$$Speed-up = \frac{T(1)}{T(N_p)} \quad (8)$$

$$Efficiency = \frac{Speed-up}{N_p} \times 100\% \quad (9)$$

where N_P is the number of processors, $T(N_P)$ is the computational time for the parallelized calculations by N_P processors, and $T(1)$ is

Table 1. The computing time calculated by Fastlap for different numbers of panels.

Partitioning depth	$d = 2$	$d = 3$	$d = 4$
3136 panels	421.47 s	23.140 s	3.671 s
4096 panels	2350.9 s	58.412 s	6.447 s
4900 panels	4964.9 s	117.161 s	9.988 s

Table 2. The computing time of each step in low partitioning depths.

N3136	$d = 2$	Percentage	$d = 3$
Total time:	421.47 s	100.00%	23.14 s
Data structure setup time:	0.18 s	0.04%	0.17 s
Direct matrix setup time:	1.8 s	0.43%	0.75 s
Multipole matrix setup time:	0.31 s	0.07%	0.87 s
P*q product time, direct part:	2.2 s	0.52%	0.86 s

Figure 6. The speed-up vs. partitioning depths for different number of panels. (a), (b) and (c) show the speed-up by the approaches described in Subsections 3.1, 3.2 and 3.3, respectively. Note that the number of panels for these cases is 4900.

the computational time by the original Fastlap (un-parallelized, one processor).

The best speed-up occurs at the lowest partition depth ($d = 2$). Also, the speed-up by the approach described in Subsection 3.1 is not very efficient (efficiency is less than 60%), while the other two parallelization approaches (described in Subsections 3.2 and 3.3)

Figure 7. Schematic view of an emitter. The solid lines (S_g) are Dirichlet boundaries ($\psi=\text{const}$), the dashed lines (S_s) are Neumann boundaries for symmetry ($\frac{d\psi}{dn} = 0$), and dotted line (S_e) is Neumann boundary for the uniform field ($\frac{d\psi}{dn}=\text{const}$).

give similar results (about 70% efficiencies). Notice that as the partitioning depth increases, speed-up (efficiency) decreases rapidly. This is because at a high partitioning depth, the size of each submatrix becomes smaller, while the total number of those submatrices increases exponentially. Therefore, the required data transmission time through the local network of the PC cluster becomes dominant when compared with the cost of inverting submatrices, which in turn makes the parallelization much less efficient. It is obvious that the calculated results by the parallelized and un-parallelized (original) Fastlap are identical. In summary, this simulated case indicates that the computing time can be effectively reduced without any compromise in accuracy for the simulations with low partitioning depths.

4.2. Case II: A Field Emission Device

The most significant challenge in the numerical modeling for field emission devices (FED) is that the dimensional scales of the tip region and the region around the gate differ by about 3 orders of magnitude [14]. For example, the typical FED tip radius of curvature is about $5 \sim 10$ nm and the dimension of the gate is about $5 \mu\text{m}$. Therefore, significant computing time is usually required because of the high density of BEM panels around the tip region. Fig. 7 shows the boundary element model of a field emission device. The solid lines are the Dirichlet boundaries, the dashed lines are the Neumann

Figure 8. (a) 3-D boundary element mesh of a field emission device. (b) Side view of the 3-D boundary element plot. A high density of panels exists in the tip region.

Figure 9. The speed-ups and efficiencies for the FED case.

boundaries of symmetry, and dotted line is Neumann boundary for the uniform field [20]. Fig. 8 shows the 3-D BEM surface mesh for a single field emitter structure. Note that since the symmetric boundaries (Neumann boundaries) are used (the dashed lines in Fig. 8), the electric field or the potential simulated by the model shown in Fig. 8 are in fact for a 2D array of FED tips. The detailed dimensions and the comparison with measured results can be found in [14]. Fig. 9 illustrates the speed-ups and the efficiencies vs. the number accuracy. By using the approach described in Subsection 3.3, the

highest efficiency (about 90% efficiency in eight nodes) is obtained. On the other hand, the approaches described in Subsections 3.1 and 3.2 are not optimal for this case because of the highly non-uniform distribution of the BEM panels. The efficiencies for these two approaches also drop rapidly when the number of computing nodes increases.

5. CONCLUSION

In this paper, we present a methodology for parallelizing preconditioned electrostatic BEM solvers that employ the multipole or the FFT algorithms. The parallelization computations are performed on an eight-node PC cluster. In order to efficiently generate the BEM preconditioner matrix, the submatrices, which are extracted from the original BEM system matrix, are distributed to the nodes of the PC cluster for inversion calculating. Three different parallel approaches for distributing submatrices are presented. A cost model that accounts for the computation cost and the network transmission cost is also proposed for balancing the load. Simulated results show that our parallel approaches provide good efficiencies without any compromise in accuracy. It must be emphasized that for the condition that the distributions of BEM panels are non-uniform, the parallelization approach based on the proposed cost model provides an extraordinarily excellent speed-up (about 7.25 for a 8-node cluster for the FED case, over 90% efficiency). Especially, the advantage of the proposed approaches is very effective for low partitioning-depth cases which usually provide the most accurate results.

ACKNOWLEDGMENT

The author would like to thank Professor W.-F. Wu for supporting this work and Mr. D.-W. Lin for helping construct the PC cluster. This work was supported in part by the National Science Council, Taiwan, R.O.C. under Contract No. NSC 92-2213-E-002-083.

REFERENCES

1. Hall, W. S., *The Boundary Element Method*, Kluwer Academic Publishers, Boston, 1994.
2. Atkinson, K. E. and M. A. Golberg, "A survey of boundary integral equation methods for the numerical solution of Laplace's equation in three dimensions," *Numerical Solution of Integral Equations*, 1-34, Plenum Press, New York, 1990.

3. *CoventorWare 2001 Reference Manual*, Coventor, Inc., 2001.
4. Korsmeyer, F. T., K. S. Nabors, and J. K. White, *Fastlap Version 2.0*, Massachusetts Institute of Technology, Cambridge, MA, 1996.
5. Phillips, J. R. and J. K. White, "A precorrected FFT method for electrostatic analysis of complicated 3-D structures," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 10, 1059–1072, Oct. 1997.
6. Greengard, L. and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, Vol. 73, No. 2, 325–348, Dec. 1987.
7. Buchau, A. and W. M. Rucker, "Pre-conditioned fast adaptive multipole boundary element method," *IEEE Transactions on Magnetics*, Vol. 38, No. 2, 461–464, 2002.
8. Nabors, K. S., F. T. Korsmeyer, F. T. Leighton, and J. K. White, "Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory," *SIAM Journal of Scientific Computing*, Vol. 15, No. 3, 713–735, May 1994.
9. Saad, Y. and M. H. Schultz, "GMRES: A generalized minimum algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comp.*, Vol. 7, No. 3, 856–869, July 1986.
10. Aluru, N. R., V. B. Nadkarni, and J. White, "A parallel precorrected FFT based capacitance extraction program for signal integrity analysis," *Proceeding of ACM/EEE Design Automation Conference*, Jun. 1996.
11. Li, L.-W., Y.-J. Wang, and E.-P. Li, "MPI-based parallelized precorrected FFT algorithm for analyzing scattering by arbitrarily shaped three-dimensional objects," *Journal of Electromagnetic Waves and Applications*, Vol. 17, No. 10, 1489–1491, Oct. 2003.
12. Wang, Z., Y. Yuan, and Q. Wu, "A parallel multipole accelerated 3-D capacitance simulator based on an improved model," *IEEE Trans. Computer-Aided Design*, Vol. 15, 1441–1450, Dec. 1996.
13. Yuan, Y. and P. Banerjee, "A parallel implementation of a fast multipole based 3-D capacitance extraction program on distributed memory multicomputers," *Journal of Parallel and Distributed Computing*, Vol. 61, No. 12, 1751–1774, Dec. 2001.
14. Yang, Y.-J., F. T. Korsmeyer, V. Rabinovich, M. Ding, S. D. Senturia, and A. I. Akinwande, "An efficient 3-dimensional CAD tool for field-emission devices," *IEDM 1998*, San Francisco, Dec. 1998.
15. Yang, Y.-J., "Numerical analysis and design strategy for field-

- emission devices,” Doctoral Dissertation in Electrical Engineering, MIT, 1999.
16. Sambavaram, S. R., V. Sarin, A. Sameh, and A. Grama, “Multipole-based preconditioners for large sparse linear systems,” *Parallel Computing*, Vol. 29, No. 9, 1261–1273, Sep. 2003.
 17. Anderson, T. E., D. E. Culler, and D. A. Patterson, “A case for networks of workstations: NOW,” *IEEE Micro.*, Vol. 15, No. 1, 54–64, Feb. 1995.
 18. Bruck, J., D. Dolev, C.-T. Ho, M.-C. Rosu, and R. Strong, “Efficient message passing interface (MPI) for parallel computing on clusters of workstations,” *Journal of Parallel and Distributed Computing*, Vol. 40, No. 1, 19–34, Jan. 1997.
 19. Greenbaum, A., L. Greengard, and G. B. McFadden, “Laplace’s equation and the dirichlet-neumann map in multiply connected domains,” *Journal of Computational Physics*, Vol. 105, No. 2, 267–278, Apr. 1993.
 20. Greengard, L., *The Rapid Evaluation of Potential Fields in Particle Systems*, M.I.T. Press, Cambridge, Massachusetts, 1988.
 21. Tang, H.-K. and Y.-J. Yang, *Parallelization of BEM Electrostatic Solver Using a PC Cluster*, Nanotech, Boston, USA, March, 2004.

Yao-Joe Yang received the B.S. degree from the National Taiwan University, Taipei, Taiwan, in 1990, and the M.S. and the Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1997 and 1999, respectively. From 1999 to 2000, he joined the Coventor Inc, (Cambridge, MA) as a senior application engineer. Since 2000, he joined the Department of Mechanical Engineering at the National Taiwan University, Taipei, Taiwan. Currently he is an assistant professor. He also serves as the director for CAD Technology in the Northern NEMS Center sponsored by the National Science Council, Taiwan. His research interests include microelectromechanical systems, nanotechnology, parallel processing, and semiconductor devices and vacuum microelectronics modeling. Dr. Yang is a member of IEEE.

Hua-Kun Tang was born on March 4, 1980 in Taoyuan, Taiwan. He received the B.S. degree and is currently working toward the M.S. degree at the department of mechanical engineering at the National Taiwan University, Taipei, Taiwan. His research interests include parallel processing, computational mechanics, and numerics.