An Interactive Remote Visualization Environment for an Electromagnetic Scattering Simulation on a High Performance Computing System

Gang Cheng[†‡], Yinghua Lu[†], Geoffrey Fox[†‡], Kim Mills[†] and Tomasz Haupt[†]

[†]Northeast Parallel Architectures Center
[‡]School of Computer and Information Science
Syracuse University, Syracuse, NY 13244

Abstract

Electromagnetic scattering(EMS) simulation is an important computationally intensive application within the field of electromagnetics. Advances in high performance computing and communication (HPCC) and data visualization environment(DVE) provide new opportunities to visualize real-time simulation problems such as EMS which require significant computational resources. In this work, an integrated interactive visualization environment was created for an EMS simulation, coupling a graphical user interface(GUI) for runtime simulation parameters input and 3D rendering output on a graphical workstation, with computational modules running on a parallel supercomputer and two workstations. Application Visualization System(AVS) was used as integrating software to facilitate both networking and scientific data visualization. Using the EMS simulation as a case study in this paper, we explore the AVS dataflow methodology to naturally integrate data visualization, parallel systems and heterogeneous computing. Major issues in integrating this remote visualization system are discussed, including task decomposition, system integration, concurrent control, and a high level DVE-based distributed programming model.

1 Introduction

Scientific visualization has traditionally been carried out interactively on workstations, or in post-processing or batch on supercomputers. With advances in high performance computing systems and networking technologies, interactive visualization in a distributed environment becomes feasible. In a remote visualization environment, data, I/O, computation and user interaction are physically distributed through high-speed networking to achieve high performance and optimal use of various resources required by the application task. Seamless integration of high performance computing systems with graphics workstations and traditional scientific visualization is not only feasible, but will be a common practice with real-time application systems [18, 16, 15].

Electromagnetic scattering(EMS) simulation represents an important computationally intensive application in industry, and is an area of emphasis in the national high performance computing initiative[5]. In previous work, an electromagnetic scattering(EMS) problem was used as an application problem in a benchmark suite for the Fortran-90D/High Performance Fortran development at Northeast Parallel Architectures Center(NPAC) at Syracuse University, to evaluate parallel algorithm and programming language issues of this application on parallel systems [14]. To further develop this application and provide engineers with visual insights into EMS simulation problem, an interactive remote visualization environment was developed. Over a Ethernet-based local network, this environment combines a graphical user interface of runtime system control and 3D graphics rendering on a graphical workstation with parallel and sequential computational modules running on a parallel supercomputer Connection Machine CM5 and two SUN Sparc stations. Application Visualization System(AVS) was used in this real-time simulation system as a data visualization environment(DVE), enabling high level 3D data visualization and networking capabilities.

In this paper, we address a number of issues encountered in building an interactive visualization environment for EMS, including decomposition of computation, system integration, interstage communication and concurrent control, and a high level DVE-based distributed programming model.

2 The Electromagnetic Scattering Problem

Electromagnetic scattering is a widely encountered problem in electromagnetics [7, 10, 21], with important applications in industry such as microwave equipment, radar, antenna, aviation, and electromagnetic compatibility design. Figure 1 illustrates the EMS problem we are modeling. Above an infinite conductor plane, there is an incident EM field in free space. Two slots of equal width on the conducting plane are interconnected to a microwave network behind the plane. The microwave network represents the load of waveguides, for example, a microwave receiver. The incident EM field penetrates the two slots which are filled with insulation materials such as air or oil. Connected by the microwave network, the EM fields in the two slots interact with each other, creating two equivalent magnetic current sources in the two slots. A new scattered EM field is then formed above the slots. We simulate this phenomena and calculate the strength of the scattered EM field under various physical circumstances. The presence of the two slots and the microwave load in this application requires simulation models with high performance computation and communication. Visualization is very important in helping scientists to understand this problem under various physical conditions.

In previous work, data parallel and message passing algorithms for this application were developed to run efficiently on massively parallel SIMD machines such as Connection Machine CM-2 and DECmpp-12000, and MIMD machines such as the Connection Machine CM-5 and iPSC/860. The data parallel algorithms run approximately about 400 times faster than sequential versions on a high-speed workstation [11]. Parallel models on high performance systems provides a unique opportunity to interactively visualize the EMS simulation in real-time. This problem requires response time of a simulation cycle that is not possible on conventional hardware.

3 Parallization of the EMS Simulation Model

3.1 A Distributed Computing Model

In [3], a simple and feasible performance model of a remote visualization environment for a financial modeling application was developed. This model can be generalized as follows, shown in Figure 3 in which distributed I/O is not considered. A simulation cycle is started from a GUI module running on the local machine, with simulation parameters are represented as slide buttons and dials to provide real-time instrumentation of the simulation's progress as well as simulation steering (changing simulation parameters before a new simulation cycle). Source data for the simulation can also be read in by the GUI module from databases or data files. The computational task of the simulation is decomposed into a set of computationally relatively independent subtasks, which are represented as computing modules distributed to different remote machines. Input data are collected from the GUI by user runtime interaction, from disk files, or both sources, and broadcasted to computing modules on the remote machines. There is no data transfer among modules on different remote machines. A remote machine may be a workstation or a supercomputer, whichever architecture and computational power best suited to the decomposed subtask. The simulation ends with some kind of GUI rendering/viewing modules that run on the local machine and use results generated from remote modules. Fundamentally, this is a dataflow (data-driven) programming model, in which activation of a module process is triggered solely by availability of input data from another module process on either the same or a different machine.

This general model is well suited to rapid-prototyping certain simulation and modeling applications that require both scientific data visualization and high performance computing. At the software environment level, this model only requires support of high level data visualization and networking facilities. Most importantly, this dataflow based model is well supported by most commercially available DVEs, such as AVS[1] from AVS Inc. and Explorer[17] from Silicon Graphics Inc..

Components of our distributed model for simulation and visualization are:

1. Decomposition of visualization computation,

2. Selection of simulation parameters,

3. Design of a graphical user interface,

4. System integration and synchronization, and
5. Interstage communication and performance analysis.

Using the EMS simulation, we next examine decomposition of visualization computation and selection of simulation parameters. The remaining components are discussed in later sections.

3.2 Computational Task Decomposition

Control parallelism and data parallelism are used to target at the distributed computing model illustrated in the preceding section and a MIMD parallel machine along with a number of high-speed workstations. We use the following guidelines for the decomposition of our model's control parallelism:

1. Hardware architecture and computational power best suited to decomposed subtasks.
2. Logical components of the application's computational model.
3. Performance balance among decomposed modules, and
4. Communication requirements.

The moment method [8, 9, 12] is used as the numerical model for the EMS problem, which can be represented as:

$$\{[\ Y^a\ ] + [\ Y^b\ ]\}\vec{V} = \vec{I}$$

$$[\ H\ ] = \mathcal{L}\{f\ (\vec{V}, \vec{M}, [\ H_0^2\ ])\}$$

where,
$[\ Y^a\ ]$: equivalent admittance matrix of the free space;
$[\ Y^b\ ]$: equivalent admittance matrix of the microwave network;
$\vec{V}$: coefficient vector;
$\vec{I}$: the excitation vector;
$\vec{M}$: a vector of mode functions;
$[\ H_0^2\ ]$: matrix of Hankel functions;
f : a function;
$\mathcal{L}$: a linear operator on f;
$[\ H\ ]$: final matrix of the simulated EMS field strength.

From the previous parallel algorithm design, we observed that:

1. Calculations of $[\ Y^a\ ]$, $[\ Y^b\ ]$, $\vec{I}$, $\vec{M}$, and $[\ H_0^2\ ]$ can be done independently;

2. Computation of $[\ Y^a\ ]$, $[\ H_0^2\ ]$, and the linear solver for $\vec{V}$ have significant communication requirements and are computationally intensive;

3. $[\ Y^b\ ]$ is a sparse matrix and calculation of $\vec{M}$ requires little time. Calculation time for $[\ Y^a\ ]$, $[\ Y^b\ ]$ and $\vec{I}$ are relatively balanced.

Thus, we partition computations of this application into four loosely coupled computing modules(they are named as 'EM-1-SUN', 'EM-2-SUN', 'EM-3-CM5' and 'EM-all-CM5' in the AVS module network of Figure 6). Three modules can run simultaneously in the distributed computing environment(see Figure 4).

At the second level, i.e., data decomposition, because most computations of $[\ Y^a\ ]$, $\vec{V}$(a linear solver), $[\ H_0^2\ ]$, and $[\ H\ ]$ are matrix manipulations, data parallel algorithms are developed in Fortran90 and tailored to run on the Connection Machine 5, to take the advantages of CM5's balanced data network and control network. The CM Scientific Subroutine Library (CMSSL) is used in the data parallel implementation[13].

At a more general level, we can view the entire system a 'metacomputer' that makes use of both functional parallelism and pipelining. In this application, functional parallelism consists of graphical I/O (i.e., user interaction, 3D rendering) and decomposed simulation computations which are handled concurrently by different components of the metacomputer. Pipelining combines calculations and communications among different processors or groups of processor(e.g. the CM5) that are carried out simultaneously in consecutive stages of the simulation. We will discuss pipelining execution of the system in later sections.

3.3 Selection of Simulation Parameters

Simulation parameters are implemented as control widgets within a graphical user interface to provide a visual medium for the user to interact with the simulation and visualization at runtime. We choose parameters representing the physical foundation of the EMS problem and the computational model that the user can manipulate in order to visually understand the problem.

As shown in Figure 2, the following parameters are graphically represented to allow user's runtime input:

1. Φ : angle of the incident EM field(wave);
2. i : $i \times i$ is the simulated area of the EMS field;
3. w : width of the slot;
4. d : distance between the two slots;
5. ϵ : permittivity of the media in the slot;
6. μ : permeability of the media in the slot;
7. h : height of the slots;
8. $y_{11}, y_{12}, y_{21}, y_{22}$: admittance parameters of a given microwave network.

In addition, there are four parameters representing characteristics of the equivalent magnetic current originally formed by the penetrating incident EM field. These parameters are used as mode function expansions and the number of pulse functions in the moment method. Another parameter used for visualization purposes is the number of grid points for discretizing the visualized EMS area. All the five parameters have direct impact on the computational requirements and simulation resolution of the moment method. Using different combinations of the parameters, we can visualize the EMS simulation under a large number of real application circumstances. We also use visualization of typical parameters to verify the EMS computational model and some well-known theories about this physical phenomena.

4 Implementation of the Remote Visualization Environment

4.1 System Configuration and Integration Using AVS

Figure 5 illustrates the system configuration and module components distributed over the network to three high-end workstations and a supercomputer Connection Machine 5. The network is a 10 MBit/s Ethernet-based local network. Commercially available AVS software is used to provide sophisticated 3D data visualization and system control functionality required by the simulation. We use AVS to facilitate high level networking and data transfers among visualization and computational modules on different machines in the system. AVS provides a data-channel abstraction that transparently handles type-conversion and module connectivities. This software system is optimized for data movement by using techniques such as shared memory message passing among modules on the same machine. Message passing occurs at a high level of data abstraction in AVS. This approach helps to make optimal use of both the high performance computing resources and the rendering capabilities of the local graphical workstation. The transparent networking capabilities of AVS open up possibilities for visualization far beyond traditional graphics capabilities[1, 2].

The local machine in our system is a IBM RS/6000 with a 24-bit color GTO graphics adaptor. An AVS coroutine module (in C) on the local machine serves as a graphical input and system control interface to monitor and collect user runtime interaction with the simulation through keyboard, mouse and other I/O devices. The AVS kernel also runs on the local machine, coordinating data flows and control flows among AVS (remote) modules in the network.

We use an AVS system module called 'geometry viewer' along with other system modules('generate colormap', 'color range' and 'field to mesh') for 3D rendering operations.

The computationally intensive modules of this application are distributed to the CM5[20], a MIMD supercomputer which is configured 32 processing nodes at NPAC. Each processing node(PN) of the CM5 consists of a SPARC processor for control and non-vector computation, four vector units for numerical computation and 32 MB of RAM. It also includes a Network Interface chip which gives the node access to the CM5 internal Data Network and Control Network. The two internal networks connect all the PNs with a control processor(CP) which runs a custom version of SunOS on a SPARC host.

Two Sun SPARC workstations are used in our distributed visualization environment to run the computational modules with modest communication requirements.

All modules other than those on the local machine are implemented as AVS remote modules. Their input/output ports are defined by specific AVS libraries for receiving/sending data from/to other (remote) modules via socket connections. This configuration allows the interrupt driven user interface input mechanisms and rendering operations to be relegated to the graphical workstation, while the computationally intensive components run on the CM5 coupled with the two workstations. This distributed simulation environment implemented in AVS provides a transparent mechanism for using distributed computing resources along with a sophisticated user interface component that permits a variety of interactive, application-specified inputs.

The flow-chart diagram in Figure 6(lower right) is a module network configured by the AVS Network Editor for this application.

4.2 The Graphical User Interface

The graphical user interface includes a main control panel, three individual input panels, and a 3D rendering window.

The main control panel provides the user parameters input and simulation control at runtime. There are seven dial widgets representing simulation parameters used by all computing modules on the three remote machines, and a control button for starting a new simulation cycle(see lower left in Figure 6). The rest of simulation parameters discussed in Section 3.3 are implemented as dial widgets in individual panels associated with the modules only requiring them, to minimize redundant data transfers between the input interface module on the local machine and computational modules on remote machines. They can also be turned on from the AVS Network Control Panel(not shown in Figure 6).

Using the AVS Geometry Viewer, 3D simulation data can be rendered in various forms such as move, rotate, scale, move the eye point and perspective view, and with sophisticated rendering techniques such as lighting and shading, multiple camera, Z-buffering, 2D and 3D texture mapping, automatic removal of hidden surfaces, sphere rendering, etc. The geometry viewer also takes advantage of the hardware rendering capabilities of the GL library and the GTO graphics adaptor of the IBM RS/6000. Figure 6(upper) shows a typical AVS Geometry Viewer window with two cameras from two different angles.

In addition, the AVS Network Editor provides a visual programming interface enabling the user to interactively reconfigure and reuse network modules. The Layout Editor in AVS allows easy and complete customization of the control panels in the GUI. Figure 6 illustrates this interface.

4.3 Interstage Communication and Synchronization

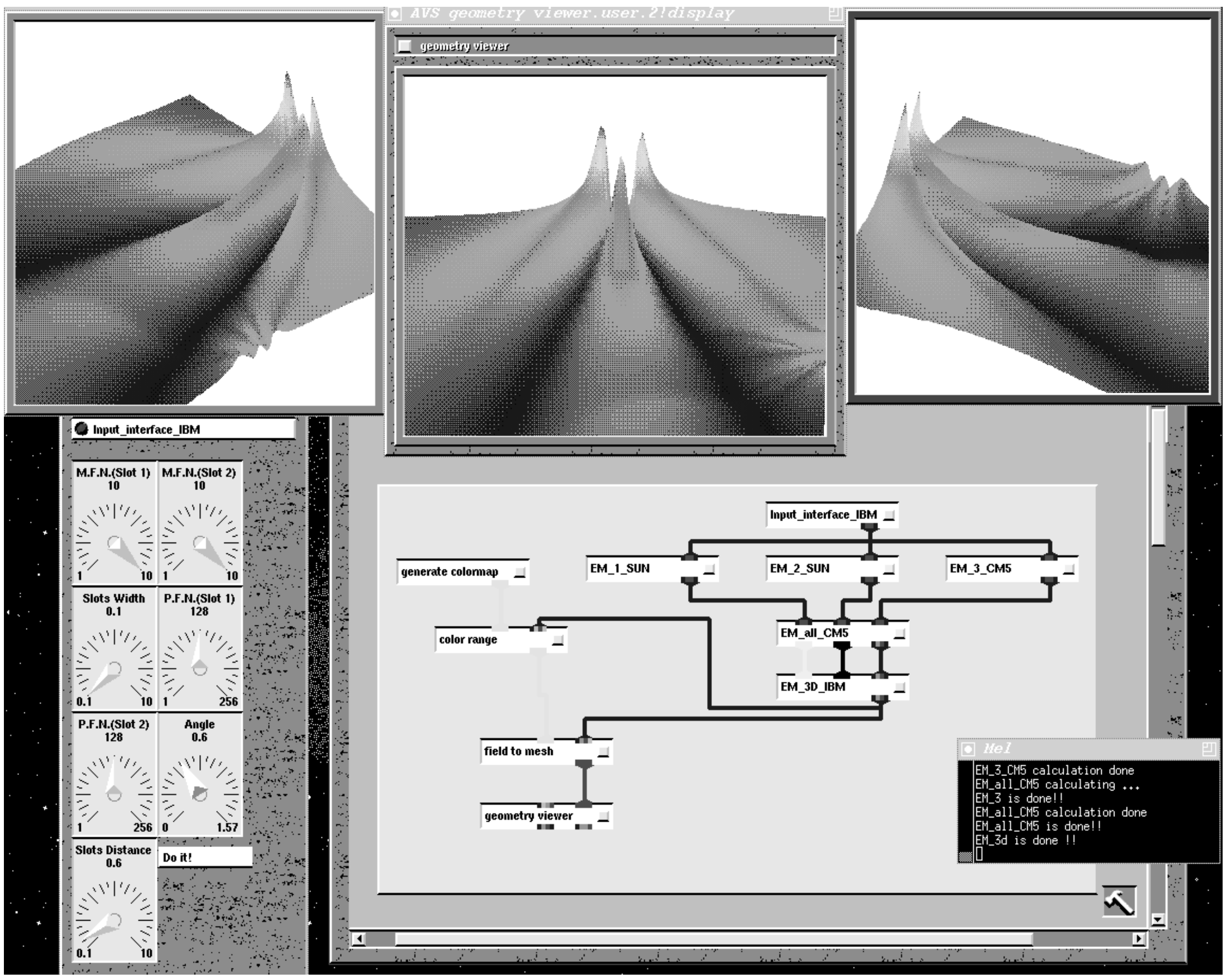
Using the data-flow programming model in AVS, message passing among modules on the same machine and on different machines are identical and completely transparent to module programmer. AVS kernel(protocol) supervises data transfer which is eventually carried out by TCP/IP at a lower level. The module programmer needs only to define module input and output ports in AVS predefined data types. Message passing among AVS modules occurs only through I/O ports. A set of routines for initializing and describing modules to AVS, as well as parameter handling, accessing data, error handling and coroutine event handling are provided. Data sources and destinations can be flexibly defined by visually connecting module input and output ports using the Network Editor.

In most cases, we use an AVS 'field' as the transferred data type which is actually a C structure. In the module network shown in Figure 6, data transmissions are overlapped with simulation computations(pipelining). For example, the computation of 'EM-3-CM5' can be pipelined with the data transfer between 'Input-interface-IBM' and 'EM-2-SUN' or 'EM-1-SUN'. Message passing within a machine is implemented in AVS by copying pointers to the same memory (shared memory). Thus, there is no network communication cost for transferring the matrix $\left[\ Y^b\ \right]$ between the two modules, i.e.,'EM-3-CM5' to 'EM-all-CM5', on CM5's control processor. Instead of generating a complete 3D data on the CM5 and sending them to rendering modules on the local machine, we use a computing module 'EM-3D-IBM' on the local machine to generate the X-Y coordinate data. Only the Z-coordinate data(i.e., $\left[\ H\ \right]$) and two scalar parameters for defining the X-Y data are transferred from the CM-5 to the local machine.

Simulation parameters are defined as input ports and connected to widgets on control panels supervised by the X-window manager on the local machine. By distributing parameter only to the modules that require them, we minimize recomputation of modules in a new simulation cycle. For example, changing of a parameter on the control panel of 'EM-1-SUN' will not activate the 'EM-3-CM5' and 'EM-2-SUN'.

This system is designed to work under a complete resource and time sharing environment(shared Ethernet, CM5 and all remote workstations) thus wall-clock time for computation and communication of all modules is difficult to predict. Concurrent control in the module network plays an extremely important role in assuring correctness, robustness and reliability. We issue concurrent controls in three different places:

1. In the main control panel of the GUI, a one-shot control button is set to allow the end-user to control the start of a new simulation cycle.
2. Broadcast of parameters from the module 'input-interface-IBM' to the computing modules on the



```
EM_3_CM5 calculation done
EM_all_CM5 calculating ...
EM_3 is done!!
EM_all_CM5 calculation done
EM_all_CM5 is done!!
EM_3d is done !!
```

Figure 6: The Graphical User Interface on the Local Machine

remote machines is performed only when all the other module processes in the network are inactive.

3. Module 'EM-all-CM5' is implemented as a coroutine such that computations of the linear solver for $\vec{V}$ and $[\ H\ ]$ will not be activated until all the required input data for $[\ Y^a\ ]$, $[\ Y^b\ ]$ and $\vec{I}$ have been recieved ($[\ Y^a\ ]$ if only parameters on control panel of 'EM-1-SUN' are changed).

4.4 Experimental Results

A preliminary performance requirement analysis of this general performance model shown in Figure 3 can be found in [3]. Our experiments show that under a typical working environment(only 0.5 MBits/s of the Ethernet's 10 MBits/s capacity are available), a complete simulation cycle for a set of typical EMS parameters takes about 8 seconds. This response time is quite satisfactory for this application. Table 1 lists timing data of major system components. For comparison, timings of sequential implementation on a SUN4 workstation of the two parallel modules are also given in the table.

5 Conclusion and Future Work

The performance limiting factors in this system are the sequential rendering operations on the local machine, and high-latency data transfer over the local area network due to multiple communication protocol layers. We focus here on the feasibility of applying a high-level distributed programming environment to a real application problem which requires both sophisticated 3D data visualization and high performance computing. This work shows that the DVE like AVS can be used not only for data visualization tasks (as primarily with on uniprocessor machines), but also as a general-purpose high level distributed programming tool. We will further examine this approach and compare this data-flow model with those employed by Fortran M[4], CC++ and PVM[19].

Future work in this environment will integrate some low-level message passing mechanisms to allow more flexible and multiple message passing programming paradigms. For example, we could use PVM[19] or FortranD[6] to implement one of the parallel AVS modules on networked workstations. We will also investigate the feasibility and issues concerned with developing a DVE-based programming environment on a MIMD machine. For instance, on a CM5, an extended DVE kernel can take advantages of CM5's high-bandwidth and low-latency internal networks, while many system modules(e.g. 3D rendering) are developed as parallel modules(similar work in CMAVS is being under development at Thinking Machines Corporation[15]). We view this data-flow model as a general, high-level programming environment which integrates data parallelism with control parallelism, and sequential programming with parallel programming.

References

[1] Advanced Visual Systems Inc. *AVS 4.0 Developer's Guide*, May 1992.

[2] Advanced Visual Systems Inc. *AVS 4.0 User's Guide*, May 1992.

[3] G. Cheng, K. Mills and G. Fox, *An Interactive Visualization Environment for Financial Modeling on Heterogeneous Computing Systems*, Proc. of the 6th SIAM Conference on Parallel Processing for Scientific Computing, March 1993, Norfolk, VA.

[4] I. Foster and K. M. Chandy, *Fortran M: A Language for Modular Parallel Programming*, Preprint MCS-P237-0992, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1992.

[5] G. Fox, *Parallel Computing in Industry: An Initial Survey*, in Proc. of Fifth Australian Supercomputing Conference, World Congress Centre, Melbourne, Australia, December, 1992.

[6] G. Fox, S. Hiranadani, K. Kennedy, C. Koelbel, U. Kremer, C-W Tseng, and M-Y Wu, *Fortran D*

Language Specification, Syracuse Center for Computational Science-42c, Rice COMP TR90-141, 37 pps, 1991.

[7] R. F. Harrington, *Time-Harmonic Electromagnetic Fields*, *McGraw-Hill Book Company*, New York (1961).

[8] R. F. Harrington, *Field Computation by Moment Methods*, *the Macmillan Co.*, New York (1968). Reprinted by *Krieger Publishing Co.*, Malabar, FL (1982).

[9] R. F. Harrington, *Matrix Methods For Field Problems*, *Proc. IEEE*, vol. 55, No. 2, pp. 136-149, Feb. 1967.

[10] E. C. Jordon and K. G. Balmain, *Electromagnetic Waves and Radiating Systems*, Second Edition, *Prentice-Hall, Inc.*, Englewood Cliffs, New Jersey (1969).

[11] Y. Lu, A. G. Mohamed, G. Fox and R. F. Harrington, *Implementation of Electromagnetic Scattering from Conductors Containing Loaded Slots on the Connection Machine CM-2*, Proc. of the 6th SIAM Conference on Parallel Processing for Scientific Computing, March 1993, Norfolk, VA.

[12] Y. Lu and R. F. Harrington, *Electromagnetic Scattering from a Plane Conducting Two Slots Terminated by Microwave Network(TE Case)*, Technical Report, TR-91-2, ECE Department, Syracuse University, August 1991.

[13] Y. Lu, A.G. Mohamed, R.F. Harrington, *Implementation of Electromagnetic Scattering From Conductors containing Loaded Slots on the Connection Machine CM-2*, Technical Report, Syracuse Center for Computational Science 270, March, 1992, also CRPC-TR92209.

[14] G. A. Mohmad, G. Fox, G. Laszewski, M. Parashar, T. Haupt, K. Mills, Y. Lu, N. Lin, and N. Yeh, *Applications Benchmarking Set for Fortran-D and High Performance Fortran*, Technical Report, Syracuse Center for Computational Science 327, June, 1992, also CRPC-TR92260.

[15] G. Oberbrunner, *Parallel Networking and Visualization on the Connection Machine CM-5*, the Symposium on High Performance Distributed Computing HPDC-1, September, 1992, pp. 78-84, Syracuse, NY.

[16] G. M. Parulkar, et al, *Remote Visualization: Challenges and Opportunities*, in Proc. of the 2nd IEEE Conference on Visualization, San Diego, CA, October, 1991.

[17] Silicon Graphics Inc. *Iris Explorer User's Guide*, 1992.

[18] L. L. Smarr, *Scientific Visualization from inside the Metacomputer*, keynote speak, in Proc. of the 2nd IEEE Conference on Visualization, San Diego, CA, October, 1991.

[19] V. Sunderam, *PVM: A Framework for Parallel Distributed Computing*, Concurrency: practice and experience, 2(4), Dec. 1990.

[20] Thinking Machines Corporation, *The Connection Machine CM-5 technical summary*, Technical Report, Cambridge, MA, pp. 340-353, October 1991.

[21] J. Van Bladel and C. M. Butler, *Aperture Problems*, (Proc. NATO Adv. Study Inst. on Theoretical Methods for Determining the Interaction of Electromagnetic Waves with Structures,) Ed. by J. Skwirzynski, *Sythoff and Noordhoff international Publishers*, 1979.