AIAA 93-3179

Performance of a Characteristic-Based, 3-D, Time-Domain Maxwell Equations Solver on a Massively Parallel Computer

J.S. Shang and K.C. Hill
Flight Dynamics Directorate
Wright Laboratory Wright-Patterson AFB, OH

D. Calahan
University of Michigan
Ann Arbor, Michigan

AIAA 24th
Plasmadynamics & Lasers Conference
July 6-9, 1993 / Orlando, FL

Performance of a Characteristic-Based, 3-D, Time-Domain Maxwell Equations Solver on a Massively Parallel Computer

J. S. Shang* & K. C. Hill†
Wright Laboratory, Wright-Patterson Air Force Base, Ohio 45433-7913
and
D. Calahan‡
University of Michigan, Ann Arbor, Michigan

Abstract

A characteristic-based, windward numerical procedure for solving three-dimensional Maxwell equations in the time domain has been successfully ported to the Intel Touchstone Delta multicomputer. The numerical results by concurrent computation duplicated the earlier simulations of an oscillating electric dipole on a vector processor and compared well with the exact solution. The parallelized code is scalable up to 512 nodes and incurs only a 2.91% performance degradation. The sustained data processing rate is clocked at 5.704 Gigaops. However, the data I/O process is unscalable on the shared memory system.

Nomenclature

E	Electric field intensity
H	Magnetic field intensity
i, j, k	Indices of discretized grids
J	Electric current vector
L	One-dimensional difference operator
n	Index of time level
t	Time
W	One-dimensional characteristic
x, y, z	Cartesian coordinates
r, θ, ϕ	Spherical coordinates
ϵ	Electric permittivity
μ	Magnetic permeability
λ	Eigenvalue

1 Introduction

The improvement of numerical efficiency is one of the urgent needs of computational electromagnetics (CEM) in aircraft signature technology. In this area of applications, the CEM simulations are generally more computationally intensive than computational fluid dynamics (CFD) problems. A part of the reason is that for signature processing, the numerical accuracy requirement is more stringent. A desirable predictive dynamic range can be as high as 60 db over broad viewing ranges [1]. Unlike CFD simulations, no dynamic similarity laws exist for the CEM to reduce the scaling length of computations. As a consequence, wave scattering, refracting, and diffracting phenomena must be solved on the unreduced physical dimension of the scatterer. For propagating waves, the numerical resolution of wave motions at a given frequency is dictated by the minimal wavelength of the media. From previous studies [1,2], each wavelength should be supported at least by an order of ten grid points or more to achieve a suitable numerical resolution. For media with large refraction indices, this more than ten grid point per wavelength requirement translates into an astronomical data processing rate and memory size of the supporting computer systems.

The heavy demand on computer memory and speed for CEM simulation can be illustrated by a straightforward estimate. At a single incident angle, the numerically generated signature of a modern fighter configuration from a gigahertz ($10^9 Hz$) transmitter requires a total of about twenty million grid points. Since, at minimum, three coordinates, nine direction cosines or metrics of a general curvilinear frame of reference, and six components of electromagnetic field intensities are included in the formulation [2,3,4,5], a total of nearly a half trillion memory addresses must be allocated. A typical CEM code operating on a 100 Megaops (10^6) single vector processor can process data at approximately a rate of three-tenths of a microsecond per grid point per time step. At this rate, either a monostatic or a bistatic calculation will result in more than thirty hours of computing on a sixth generation supercomputer just to advance the solution to a new time level. In order to complete a signature simulation, usually multiple look angles and hundreds of time steps are required. As a result, the large computer memory requirement and solution time has rendered the use of conventional data processors for solving CEM problems impractical.

---

*Senior Scientist. Fellow AIAA

†Electronics Engineer, Signature Technology Office

‡Professor, Electrical Engineering & Computer Science Department

Numerical efficiency improvement of computational physics for electromagnetics or fluid dynamics can be realized through novel numerical algorithm developments and concurrent computing technology. In the recent year, significant progress has been made in the area of algorithm development [6,7,8]. Numerical algorithms for solving hyperbolic equations from the CFD discipline have been adopted for solving three-dimensional Maxwell equations in the time domain [2,3,4,5,6,7]. Among these, the characteristic-based algorithms are found to be most efficient and appropriate to duplicate the wave motions that are governed by the time-dependent Maxwell equations [1,2,3]. This numerical scheme, when applied to the hyperbolic partial differential equation system, has not only greatly enhanced the stability of numerical procedure, but also yielded accurate solutions by alleviating spurious wave reflections from the truncated numerical domain [3,4,5]. Although these new numerical procedures have the potential to reduce the required computing resources by allowing larger time steps and fewer discretized mesh points in CEM simulations, substantial progress in CEM for practical applications can finally be achieved only by incorporating a massively parallel computing technique to deliver the needed computing capabilities.

Concurrent computation was adopted for CFD research in the seventies. During that period, the performance of the massively parallel computers was limited. Consequently, in a short period of a few years, the concurrent computational CFD research reached a point of diminishing returns and was abandoned. Recently, through remarkable progress in microchip and interconnect data link technology, a host of single address, shared memory, and multiple address message-passing parallel computers becomes available for data processing. These scalable multi-processors or multi-computers, in theory, are capable of providing essentially unlimited computing resources for scientific simulations. However, the effective use of massively parallel computers still rests squarely on balancing the work load and keeping the communication between computing nodes to an absolute minimum [10,11]. These requirements are intrinsically related to the numerical algorithms and hardware architecture. In the present research effort, attempts are made to map a characteristic-based algorithm onto a message-passing parallel computer for computational electromagnetics.

2 Analysis

The time-dependent Maxwell equations for electromagnetic fields in free space can be written in the following form [12,13]:

$$\frac{\partial(\mu H)}{\partial t} + \nabla \times E = 0 \qquad (1)$$

$$\frac{\partial(\epsilon E)}{\partial t} + \nabla \times H = -J \qquad (2)$$

$$\nabla \cdot B = 0; \quad B = \mu H \qquad (3)$$

$$\nabla \cdot D = 0; \quad D = \epsilon E \qquad (4)$$

The above system of partial differential equations is hyperbolic, and constitutes an initial value problem. The characteristic-based fractional-step algorithms have been demonstrated to be very efficient in solving three-dimensional Maxwell equations in the time domain [3,4,5]. The basic idea of the characteristic-based methods for solving the hyperbolic system of equations is derived from the eigenvalue and the eigenvector analyses. In numerical simulation, the well-posedness requirement for initial or boundary conditions and the stability of a solving scheme are ultimately linked to the eigenvalues of the governing equations. Therefore, the characteristic-based schemes showed a drastic improvement in numerical stability and accuracy by using a windward difference formulation according to the signs of the eigenvalues [3,4,5,6]. However, the characteristic-based algorithm also has an inherent limitation in that the coefficient matrices of the governing equations, when expressed in vector form, can be diagonalized in only one dimension at a time [6,7]. Therefore, all multiple dimensional equations are split into multiple one-dimensional formulations and solved by the fractional-step or the time-splitting methods [14,15] summarized in the following formulas:

$$w^{n+2} = L_x L_y L_z L_z L_y L_x w^n \qquad (5)$$

$$L_x : \frac{\partial w_{x,i}}{\partial t} + \lambda_i \frac{\partial w_{x,i}}{\partial x} = 0 \qquad i = 1, 2, \ldots, 6 \qquad (6)$$

$$L_y : \frac{\partial w_{y,i}}{\partial t} + \lambda_i \frac{\partial w_{y,i}}{\partial y} = 0 \qquad i = 1, 2, \ldots, 6 \qquad (7)$$

$$L_z : \frac{\partial w_{z,i}}{\partial t} + \lambda_i \frac{\partial w_{z,i}}{\partial z} = 0 \qquad i = 1, 2, \ldots, 6 \qquad (8)$$

Since these one-dimensional characteristics of the Maxwell equations are completely uncoupled from each other and appear in scalar form, the system of equations is exactly the Riemann problem [5,6]. The most important feature of the present numerical procedure is that when the windward approximations are adopted, the costly matrix inversion for an implicit scheme becomes unnecessary [3,5].

In the earlier efforts [3,4,5], an implicit and an explicit fractional-step method were developed for solving the time-dependent Maxwell equations. The characteristic formulation using the windward finite difference discretization led to a nearly identical computational effort between implicit and explicit procedures. From the sign of the eigenvalues, the stencil of the second-order

accurate windward difference approximation can be easily constructed to form the one-dimensional and diagonal difference operator. Both formulations require only a single forward or backward substitution to advance the solution to the next time level. The windward difference approximation is give by

$$\frac{\partial w}{\partial x} = \frac{-3w^*_{i,j,k} + 4w^*_{i+1,j,k} - w^*_{i+2,j,k}}{2\Delta x} \quad \text{if } \lambda < 0 \qquad (9)$$

$$\frac{\partial w}{\partial x} = \frac{3w^*_{i,j,k} - 4w^*_{i-1,j,k} + w^*_{i-2,j,k}}{2\Delta x} \quad \text{if } \lambda > 0 \qquad (10)$$

where $*$ takes the values $n+1$ and n respectively for implicit and explicit procedures.

The data flow associated with time-splitting algorithms is one directional. The strategy to map these algorithms to a massively parallel computer is therefore identical. The load balancing of concurrent computation can be achieved easily by counting and adjusting arithmetic operations for each directional numerical sweep. The partition of data structure for minimum information flow are investigated by both the task partition and the domain decomposition technique.

3 Partitions of Data Structure

The partition of data structure plays a key role in achieving high parallel efficiency. On a message passing or distributed memory multicomputer system, the performance of concurrent computing is closely tied to memory bandwidth and memory latency. These system peculiarities always exist, regardless of whether the system has a cache or a direct path to memory unit that bypasses the cache. The basic criterion of the most efficient data structure of any numerical procedure is to restrict the data movement to an absolute minimum.

The Touchstone Delta multicomputer consists of a total of 576 heterogeneous nodes [16]. They are designated as numerical, service, gateway, and disk nodes to perform the computation, frame buffer, network link, and disk string functions respectively. Each of the numerical node has a peak rate of 80 single-precision and 60 double-precision Mflops, and is interconnected by a two- dimensional mesh network. On this ensemble nodal architecture, the data flow and data management lead to four different approaches for the controlling of data movements between nodes. The most elementary approach to data partition is the one-dimensional parallelization in which the outermost do loops of each numerical sweep are assigned to individual processors. The other data partition schemes include the page structure by partitioning three dimensional space into cross-sectional planes, the pencil grouping, and finally, the three-dimensional block parallelization.

A graphic depiction of these data partition schemes is given in Figure 1. Each scheme has its unique features according to the dimensions and sequence of data string assigned to the distributed microprocessors. Within each numerical node, calculations is performed before the need to access data from other numerical nodes for computations to continue. On a specific hardware architecture, coding options of numerical algorithms will have some degree of control for data movement and data management. The guideline for high parallel efficiency is simply to keep the data transfer between nodes at the absolute minimum, and take full advantage of the message-passing priority of the immediately adjacent nodes. In order to sustain the maximum concurrent computations of a particular numerical algorithm, the load balancing and data partitioning are closely interwoven. The following discussion will focus on these issues.

In the present effort, only the one-dimensional parallelization and the pencil grouping are investigated. The former scheme is the most straightforward and probably the most suitable data partition for the fractional-step, windward procedure [5]. In this scheme, all data are organized along one of the numerical sweep directions which can be viewed as a form of task partitioning. Since the number of mesh points along the chosen coordinate equals the number of nodes used, all arithmetic operations which need to be performed in each numerical node are identical. The load balancing is achieved automatically. The pencil grouping scheme was originally designed for the ADI algorithms [3,9], in which cross-derivatives of all directions are simultaneously calculated. The pencil data structure provides connectivity between computational boundaries in the direction of numerical sweep to enforce the boundary values. The minimum size of the pencil is dictated by the stencil of cross-derivative approximations. For example, the minimal of a pentadiagonal data structure is required for the windward finite difference scheme. More importantly, the pencil data structure may not need reorganization for each individual numerical sweep, thus will not incur more data movement than the one-dimensional partition [11]. However, the load balancing can still easily be achieved among all numerical nodes.

The computational domain consists of IL, JL, and KL number of discretized mesh points in the three-dimensional space. Therefore, the total number of elements in the data array is given by the product of the three one-dimensional discretized vector lengths, ($IL \times JL \times KL$). At each time level, both one-dimensional and pencil partition approaches must perform at least 492 mixed integer and real number arithmetic operations for every discretized point. For the one-dimensional par-

tition, eight message passings are performed per grid point. In all, the total number of message passing is given by the value of $2 \times (4 \times (\text{number of nodes} - 6)$. For the base-line case ($48 \times 48 \times 48$) studied, each individual message has a length of 9,216 bytes. For the pencil grouping scheme, the number of message passings depends on the size of each pencil partition.

In mapping a serial code onto the Delta using the one-dimensional parallelization scheme, the computational domain is divided into IL grid planes. Each of the grid plane contains $JL \times KL$ mesh points. The computation associated with the $JL \times KL$ mesh points of each grid plane is assigned to a numerical node. The number of nodes used, IL, equals the product of the numbers of rows and columns of the two-dimensional matrix (m, n) partition of the Delta. With the grid plane index, 'I', ranging from '1' to 'IL' and the logic node number, 'mynode', ranging from '0' to '$mn - 1$', an easy way of assigning the grid planes to the nodes is to use the logical sequence of the nodes within the Delta partition. In other words, the grid plane index 'I' is equal to 'mynode + 1'. Unfortunately, this natural sequence assignment may result in a long and unnecessary delay because the second-order windward scheme requires data transfer between five grid planes ($I-2, I-1, I, I+1, I+2$). Particularly, the message passing in Delta is row biased. If a grid plane is located at or next to a boundary of the Delta partition, the messages of this grid plane will have to be routed through 'n' or '$n - 1$' nodes before reaching its destination grid plane located in a different row of the Delta partion.

One way to avoid this delay is to use the Serpentine sequence in assigning the grid planes to numerical nodes, such that any message need not travel more than two nodes to reach its destination grid plane. The Serpentine arrangement can be easily achieved by the following program instruction:

$$I = \begin{cases} \text{mynode} + 1 & x = \text{even number} \\ (2 * x + 1) * n - \text{mynode} & x = \text{odd number} \end{cases}$$

where the row number $x = 0, 1, 2, ...m - 1$. For example, a (4,5) Delta partition with its logic node number, natural sequence number, and Serpentine sequence number is shown in table 1.

The data I/O of the one-dimensional parallelization requires 78 arithmetic operations, and 28 system calls. Most calls are either synchronous or asynchronous message passing instructions to generate a formatted output data and three unformatted graphic files. The computation time of program initiation of the present code is negligible in comparison with program execution and data I/O time, therefore it will not be recorded here.

For the pencil grouping scheme, the data flow is reduced compared with the one-dimensional partition because only the data near the surface of the pencil must be transferred to adjacent pencils. Thus, a large pencil size will reduce the overall data flow and enhance the concurrent performance. In addition, pencil grouping scheme also provides greater flexibility than the one-dimensional scheme in assigning grid points to the numerical nodes. For example, it is possible to assign all 512 processors to a relatively small grid system (e.g, $96 \times 96 \times 96$) using the pencil grouping scheme; whereas in the one-dimensional scheme, one of the grid dimensions would have to be 512. In essence, the one-dimensional parallelization is a subset of the higher dimensional data partition. Currently, research is ongoing to further develop the higher dimensional scheme. Thus, the results will be reported at a later date.

4 Scope of numerical experiments

The mapping of the fractional-step explicit and implicit codes for solving the three-dimensional Maxwell equations in the time domain onto a multi-computer was originally planned. As mentioned previously, the diagonalization of the governing equations in each of the three time-space planes leads to three uncoupled characteristic formulations. When solving them by the windward differencing approximation [3,5], the message passing requirement is identical for both the explicit and the implicit schemes. Only the boundary values of the computational domain are specified at a different temporal level to reflect their intrinsic difference. In fact, the implicit code actually outperforms its explicit counterpart on a shared memory workstation, IRIS 4D/440 VGX. The implicit and the explicit time-splitting codes, on a ($46 \times 46 \times 46$) mesh, yielded a consistent data processing rate of 2.71×10^{-4} and 2.93×10^{-4} sec per mesh point for each time step respectively. Although the implicit procedure has an unconditionally stable property in contrast to the explicit scheme for solving the Maxwell equation, the former will neither honor the physical requirement of domain of dependence for a CFL value greater than unity, nor will it generate a perfect shift condition under a special circumstance [3,5]. For signature analyses, these features are highly desirable. Thus, the explicit procedure may be preferred. For this reason, present efforts were devoted only to map the explicit code to the Intel Delta system. However, the identical parallelization technique is equally applicable to the implicit procedure for the present formulation.

All numerical solutions were processed on the Delta system. The numerical simulations were focused on a three-dimensional oscillating electric dipole. The calculated results were validated by comparing with previous numerical results from a single vector processor [7] and the exact closed form solution [12,13]. The exact so-

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

(a)

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

(b)

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16

(c)

Table 1: (a) Mynode (b) Natural sequence (c) Serpentine sequence

lution contains a singular behavior at the center of the dipole. The leading term singularity of field intensities appears as the inverse cubic power of radial distance from the dipole [13]. Therefore, it offers a serious challenge to the accuracy of the numerical simulation and the robustness of the solving procedure.

In order to put the geometrically simple but numerically demanding simulation in the appropriate perspective, the entire electromagnetic field is presented in the form of traces of the magnetic and electric intensities. In Figure 2, the magnetic field is presented at the instance when the crest of the second wave was exiting the computational domain. The traces of magnetic lines form a series of planar concentric circles perpendicular to the dipole. The azimuthal component of the magnetic field is known to contain a singularity proportional to the inverse square radial distance from the dipole. The singular behavior at the dipole however is not apparent at the chosen time instance.

The same instantaneous time exposure of the electric field is depicted in Figure 3. The orderly loops originating from the dipole and streaks around them indicate a composite field of two predominant radial and circumferential components. The asymptotic behavior of both electric field components approaches the pole as the inverse cubic power of the radial distance. These rapidly varying functions near the dipole pose a considerable demand on the accuracy and algorithmic robustness of any numerical simulation.

The scalability of the present time-splitting procedure on the Delta system was evaluated by fixing the message passing length at $9,216$ bytes each. The timing information was collected for the performances from a single node up to the full complement of nodes of the Delta. The greatly increased mesh point density and message length at $36,864$ bytes ($96 \times 96 \times 96$) and $147,456$ bytes ($192 \times 192 \times 192$) were planned for the grid resolution study. Since the startup expenditure of the present procedure is negligible (57 mixed integer and real number arithmetic operations, as well as two system calls to get information), only the execution and data I/O timing data were collected. From these data bases, the issues of scalability and parallel efficiency will be delineated.

The distributed memory (16 Mbyte/per node, $8,192$ Mbyte total) of the Delta system also provides a unique opportunity for mesh spacing refinement studies. Numerical resolution studies of a three-dimensional simulation by doubling the number of mesh points in each coordinate will result in an eight-time memory space increment. Therefore, it is rare that the numerical accuracy assessment of a three-dimensional calculation had been carried out on several levels of mesh spacing enrichment in all coordinates simultaneously. A multiple level grid spacing refinement study was conducted to determine the suitable mesh points distribution to capture the singular behavior of the dipole.

5 Performance Evaluations

The numerical results generated on the Delta system either by the one-dimensional parallelization or pencil grouping on a ($48 \times 48 \times 48$) mesh system are identical to the earlier efforts [5]. The difference between the numerical result and the analytic solution is confined within the known truncation error of the second-order accurate scheme. On the prescribed base-line mesh spacing, the numerical error is bounded within less than a fraction of one percent. Detailed and specific comparisons with the theory will be given later as part of the mesh refinement investigation.

Since the Delta system has no profile software to measure the performance of basic library functions (trigonometry and square) and the message passing latency, data processing rates were evaluated by the time elapsed during program execution and data I/O phases. Timing results were collected for 480 time steps calculations on the base-line mesh system ($48 \times 48 \times 48$). All calculations on each node were completed within 45.61 to 52.90 seconds regardless of whether the computations were conducted on 4 or 512 numerical nodes. The data I/O, on the other hand, varied widely from 0.24 to 32.20 seconds for the Serpentine sequence, and from 0.28 to 162.80 seconds for the Natural sequence. The duration of timing sampling was selected for all calculations to perform 480 steps in time, which permitted the wave to traverse twenty times the distance across the entire computational domain. The period of sampling is deemed sufficient to yield meaningful information, and the col-

lected data are repeatable. Finally, the data scattering band is limited only to 2.7 percent.

The data processing rate is computed by the total number of arithmetic operations performed during the 480 time steps divided by the maximum and the minimum time elapsed of all nodes in use. These two values bracket the range of performance among nodes. Timing information was recorded from a 4-nodes to 512-nodes simulation at an interval of doubling the number of nodes. The Flops rate is calculated by the product of numbers of iterations, arithmetic operations, total grid points, and numerical nodes, then dividing by the executing time, $(480 \times 492 \times (48 \times 48 \times 48) \times$No. of nodes) / execution time. The calculated data rate is a conservative estimate, because 47 basic library function calls and nine system calls for data movement at every time level were not taken into consideration. These 22,560 function calls (mostly were trigonometry and square root calculations), 4,320 message passing calls per node, and 32,160 integer calculations for message description can consume a significant amount of time.

The parallel efficiency or scalability of the code is defined by the time elapsed for each array of nodes used then normalized by the execution time required for four nodes. The timing basis for scaling at four nodes has yielded consistent cpu time required, and the variation among these four nodes is negligible.

The measurement of data I/O performance is straightforward. The longest and the shortest time periods required to process a formatted data file and three unformatted graphic files for the entire simulated field were recorded for each group of nodes used. The data I/O process consisted of a combination of 38 synchronous and asynchronous message passing calls, as well as 78 integer arithmetic operations at each time step. The timing information was collected from calculations from 4-node to 512- node arrays. The formatted file contained a total of 3.98 Mbytes of data, three graphic files were about a third the size of the formatted data file. The data I/O timing results from 4-node up to 512-node simulations were again normalized by the 4-node result to demonstrate the possible performance degradation of data management.

6 Discussion of Results

The comparison of the data processing rate between the serpentine and the natural sequence data passing paths is given in Table 2. The timing data is tabulated in terms of Gigaops versus the number of numerical nodes used. In the table, the Maximum and the Minimum data processing rates are included to reveal the performance difference among nodes.

The performance difference between the natural sequence and the serpentine message path arrangements begins to appear for the number of nodes beyond 128 This difference in performance may be attributable to the memory path bandwidth and memory latency of the Delta system. In the present text, the memory latency is defined as the time delay between the instance when an instruction was issued and the moment when memory units were ready to accept the request. Unfortunately, the needed knowledge base for a detailed analysis was unavailable, therefore only the observations were reported. Since the Serpentine arrangement took advantage of the nearest neighbor priority in message passing hierarchy, the performance degradation is less than that of the Natural sequence. In fact, the slowest node of a 512-node computation using the serpentine procedure required only 3.1 percent more execution time to complete the task than the fastest node. The fastest node operating on the Serpentine arrangement attained a data processing rate of 5.966 Gigaops on 512 numerical nodes. In the natural sequence arrangement, the slowest node of a 512-node computation suffers a 10.4 percent degradation compared with the fastest node. The fastest node only delivers a data processing rate of 5.412 Gigaops in this case.

Similar concurrent performance of computations on $(96 \times 96 \times 96)$ and $(96 \times 192 \times 192)$ grid systems were also observed. The data processing rate per node was maintained in the range of 12.89 to 11.65 Megaops. These data processing rates are far below the nominal performance level of the i860 microprocessor [16]. Additional performance improvement of the present numerical procedure using the Delta system is still possible.

The comparative timing results between the natural and the serpentine sequence are depicted in Figure 4. The improvement of the latter to the data processing rate of the former is clearly demonstrated. The performance degradation by the natural sequence, which did not honor the data passing priority of the nearest neighbor hierarchy, became pronounced when more than 128 numerical nodes were used. The serpentine procedure appears to be more effective for the distributed memory system. In all, an increasing disparity of the data processing rate among nodes was also noted as the number of nodes increased beyond about 64. Sustained concurrent performance of computations on a $(96 \times 96 \times 96)$ grid system was also observed, the data processing rate per node was maintained in the range of 11.65 to 11.1 Megaops.

The scalability of the present multicomputer program based on the one-dimensional parallelization is given in Figure 5. All data processing rates are normalized by the value of the 4-node simulation. A data rate anomaly was observed for the 8-node simulation fo

	Serpentine		Logic Sequence	
Number of nodes	Maximum	Minimum	Maximum	Minimum
4	0.046	0.046	0.046	0.046
8	0.092	0.092	0.092	0.092
16	0.184	0.184	0.184	0.169
32	0.368	0.368	0.369	0.369
64	0.738	0.738	0.736	0.707
128	1.472	1.472	1.445	1.430
246	2.967	2.952	2.829	2.798
384	4.490	4.413	4.105	4.028
512	5.966	5.704	5.412	5.274

Table 2: **This table needs a caption**

reasons unknown, but was included for the sake of completeness. The superior scalable property of the serpentine sequence over that of the natural sequence is obvious. The scalability of the present code up to 512 nodes is perceived within a performance degradation of less than 3.1 percent. In fact, the significant degradation only appeared when the full complement of numerical nodes was employed.

The data I/O time in seconds for recording output is presented in Figure 6. The shortest waiting time over the entire range of nodes used was less than 0.24 seconds. However, the aggregated time required of the slowest node in computation and data I/O actually determines the completion of a numerical simulation. The data I/O time increased almost linearly with the number of nodes in use. For the 512-node calculations, the serpentine sequence used 32.2 seconds while the natural sequence needed 124.4 seconds to output the same amount of data. The drastic reduction in I/O time at a factor of 3.86 further demonstrates the superiority of the serpentine sequence over the natural sequence.

Figure 7 depicts the widely varying I/O performance between nodes for the entire range of available nodes. Again the I/O performance degraded rapidly as the number of nodes in use increased. The natural sequence yielded the maximum I/O performance discrepancy among nodes. The ratio between the most and least efficient nodes was as high as 580.4. The serpentine sequence reduced the disparity to a value about 134.2. It demonstrated that the I/O performance of the Delta is not scalable. If this behavior is a common trend for all distributed memory computer systems, this deficiency shall be a pacing item for research in concurrent computing.

Owing to some uncertain peculiarities of the Delta compiler, only a limited amount of preliminary timing data from the pencil partition is obtained at the present. In short, the serpentine ordering did improve the performance over that of the natural sequence. A general performance improvement was also noted when the partition size ($JL \times KL$) is nearly equal in JL and KL. The performance degraded as one of the pencil dimensions approached unity, a clear indication that the two-dimensional partition had degenerated into the one-dimensional parallelization. The preliminary timing data of the pencil partition has revealed a parallel efficiency gain of 37% over that of the one-dimensional partition. Continuing research will be devoted to further demonstrate the potential of the higher dimensional partition scheme.

The mesh refinement investigation was forced to reduce the scope due to the presently unknown compiler peculiarities. The orginally planned mesh refinement included a finest mesh system of $(192 \times 192 \times 192)$. However, as the number of processors increased beyond 96, the disparity in execution times among processors grew to a factor of 11.6 for 108 nodes, and 14.43 for 144 nodes. The large performance discrepancy among the numerical nodes appeared only after a few numerical sweeps. In other words, the observed scalable performance of the present computer program for the cases of $(48 \times 48 \times 48)$ and $(96 \times 96 \times 96)$ was not sustainable for the mesh system of $(192 \times 192 \times 192)$. The timing ratio between the best and the worst performing nodes of the pencil partition was a modest value of 2.06, but the data was collected only after two temporal iterations.

Three mesh systems of $(96 \times 96 \times 96)$, $(48 \times 48 \times 48)$, and $(24 \times 24 \times 24)$ were used for the numerical resolution study instead. For this reduced scope of numerical experiment, the coarse mesh system has only 24 grid points in each coordinate. The mesh point density is sufficient to resolve the wave motion [1,2], but it may be deficient in simulating the singular behavior of the dipole [12,13]. Since the grid spacing was reduced by a factor of two in every coordinate from the coarsest grid to the finest grid, the initial value of the dipole

for each grid system must be specified at different grid point which corresponds to the same physical location for all three grid systems. Due to the fact that the true field is a function of time, distance from the dipole, and the wave speed [12,13], the time step must be adjusted accordingly to account for the different time step increment of each grid system.

The mesh refinement results for the radial component of the electric field are presented in Figure 8. The initial condition of the coarse mesh spacing calculation ($24 \times 24 \times 24$) was specified at the farthest distance from the dipole, where the gradients of dependent variables were diminishing rapidly. In spite of the large spacing between grid points, the local truncation errors were relatively low. The dynamic instability induced by the numerical error is actually less than the other calculations. The inherent numerical dissipation from the windward scheme [3,5] actually overwhelmed the numerical result in the immediately adjacent zone of the dipole. The subsequent mesh refinements exhibited a steady improvement, but still were insufficient to overcome the stringent demand for accurate simulation near the dipole. Nevertheless, the present numerical procedure has demonstrated the robustness in treating the problem containing a singular behavior.

Figure 9 depicts the numerical results of the circumferential component of electric intensity on the three mesh systems. This electric field component has the highest order of singular behavior of all results considered. The numerical result obtained on the finest mesh spacing ($96 \times 96 \times 96$) attained an excellent agreement with the analytic result. The maximum deviation from the theoretical results is merely two-tenths of one percent. The calculation by the ($48 \times 48 \times 48$) mesh system revealed significant numerical oscillations in an attempt to overcome the large truncation error near the dipole. In spite of the locally induced dynamic instability, the calculation was sustained for a substantial period far beyond the reported time frame. Another desired feature of the characteristic-based formulation also stood out. At a non- dimensionalized radial distance of 0.14 and beyond, all three solutions agreed well with the theory and showed no wave reflection from the truncated numerical boundary.

The comparison of magnetic azimuthal component of the three mesh system is given in Figure 10. Although the leading term singularity has a lower order asymptote than that of the electric field, the numerical behavior of solutions generated on the three mesh systems was similar. The numerical resolution produced by the finest mesh system ($96 \times 96 \times 96$), however, was able to suppress the numerical oscillation near the dipole. Finally, the reflected wave from the truncated numerical boundary was completely absent.

7 Conclusions

A fractional-step, upwind numerical procedure for solving the three-dimensional, time-domain Maxwell equations has been ported to the Intel Delta multicomputer. The concurrent computations duplicated the results from earlier numerical results and compared well with theory. For the mesh system of ($48 \times 48 \times 48$), the numerical procedure is scalable up to 512 numerical nodes with only a 2.91 percent performance degradation. The fastest data processing rate is 5.966 Gigaops and the sustained overall performance is clocked at 5.704 Gigaops. Further increased data processing rate is still possible.

The scalability of concurrent computing is sustained up to a simulation eight times the size of the baseline case. For reason uncertain, the scalable performance failed at the next level of grid point enrichment ($192 \times 192 \times 192$). Although the architecture of the Touchstone Delta multicomputer and its usefulness are impressive, consistent performance in scaling up for massive data bases remains as a necessary research emphasis.

The scalable data I/O is also identified as a pacing item for intense research for attaining high performance computation.

The one-dimension parallelization has been shown as a suitable data partition procedure for a characteristic-based, windward difference algorithm in solving the time dependent, three dimensional Maxwell equations. Further parallel efficiency improvement by pencil structure shows developable potential.

Acknowledgement The computing resource was provided by the Intel Touchstone Delta System operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access to this facility was granted by Dr G. Weigand of ARPA. Stimulating and fruitful discussions with Stephen J. Scherr and B.G. Vikstrom are also appreciated.

References

1. Anderson, D. A., Interdisciplinary Applications of Simulation: Computational Fluid Dynamics (CFD) and Radar Cross Section (RCS), AFATL-TR-88-65, Wright Laboratory, Armament Directorate, Eglin AFB FL, June 1988.

2. Shankar, V., Hall, W. F., and Mohammadian, A. H., A Time-Domain Differential Solver for Electromagnetic Scattering Problems, Proceedings of IEEE Vol. 77, No. 5, May 1989.

3. Shang, J. S., Characteristic-Based Methods for the Time-Domain Maxwell Equations, AIAA Paper 91-0606, January 1991.

4. Shang, J. S., A Characteristic-Based Algorithm for Solving 3-D, Time-Domain Maxwell Equations, AIAA Paper 92-0452, January 1992.

5. Shang, J. S., A Fractional-Step Method for Solving 3-D, Time- Domain Maxwell Equations, AIAA Paper 93-0461, January 1993.

6. Roe, P. L., Characteristic-Based Schemes for the Euler Equations, Ann Rev. Fluid Mech., Vol 18, 1986, pp 337-365.

7. Steger, J. L., and Warming, R. F., Flux Vector Splitting of the Inviscid Gasdynamics Equations with Application to Finite Difference Methods, J. Comp. Phys. Vol. 40, No. 2, February 1987, pp 263-293.

8. MacCormack, R. W., and Candler, G. V., The Solution of the Navier-Stokes Equations Using Gauss-Seidel Line Relaxation, Computer & Fluid Vol. 17, No. 1, 1989, pp 135-150.

9. Lomax, H., and Pulliam, T., A Full Implicit, Factored Code for Computing 3-D Flows on Illiac IV, Parallel Computations, Ed. G Rodriguez, Academy Press, 1982, p 217.

10. Wesley, R., Wu, E., and Calahan, D. A., A Massively-Parallel Navier-Stokes Implementation, AIAA Paper 89-1940CP, 1989.

11. Scherr, S. J., Implementation of an Explicit Navier-Stokes Algorithm on a Distributed Memory Parallel Computer, AIAA Paper 93-0063, January 1993.

12. Harrington, R. R., Time-Harmonic Electromagnetic Fields, McGraw-Hill Book Co., New York, 1961.

13. Jordan, E. C., Electromagnetic Waves and Radiation System, Prentice-Hall, Inc., NJ, 1960.

14. Bagrinovskii, K. A., and Godunov, S. K., Difference Schemes for Multi-dimensional Problems, Dokl. Akad. Nauk USSR, Vol. 115, 1957, p 431.

15. Yanenko, N. N., On Weak Approximation of Systems of Differential Equations, Sibirsk. Math. Zh., Vol. 5, 1964, p 1430.

16. Intel Corporation, Touchstone Delta System User's Guide, Publication 312125-001, October 1991.

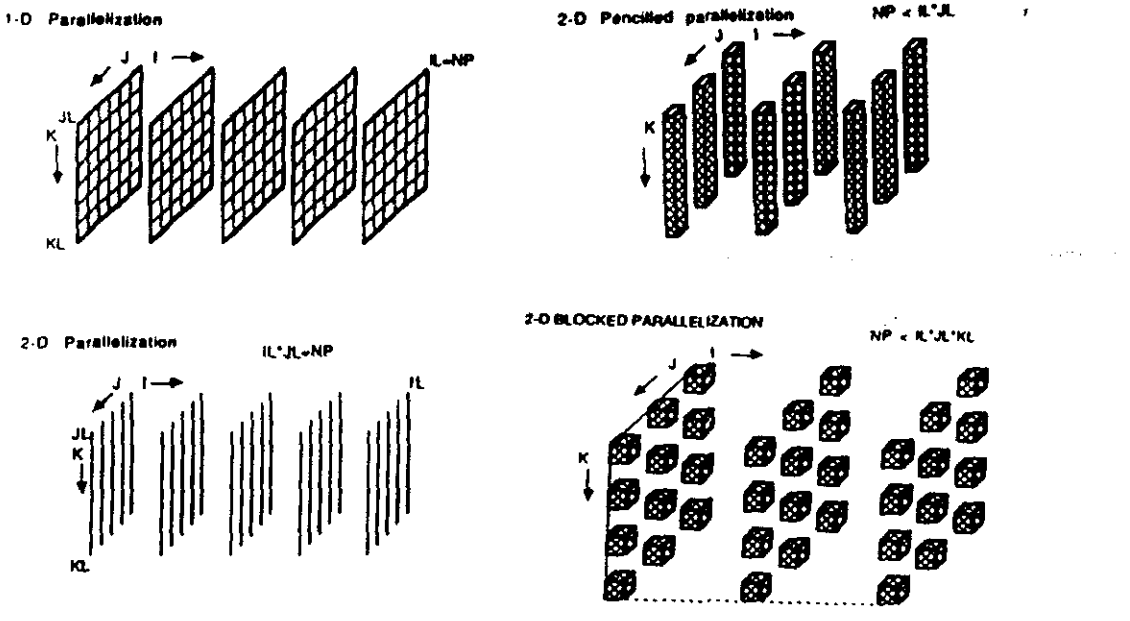

Figure 1: Hierarchy of data partition

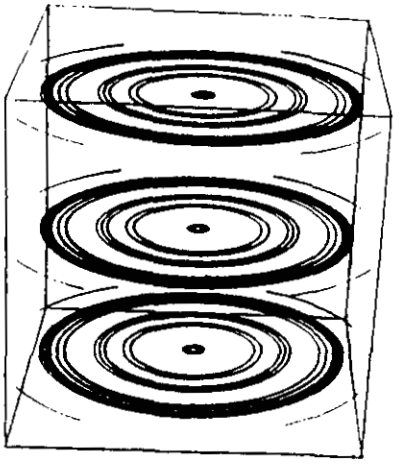

Figure 2: Magnetic field of a dipole

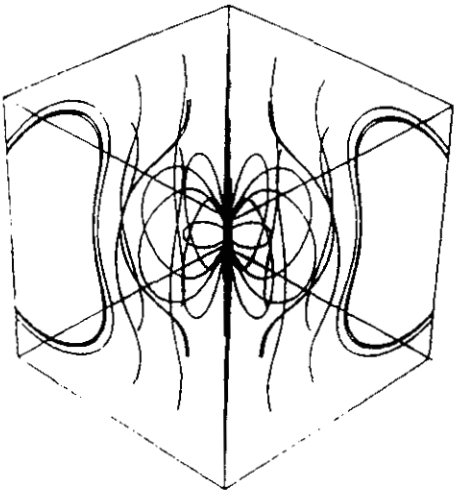

Figure 3: Electric field of a dipole

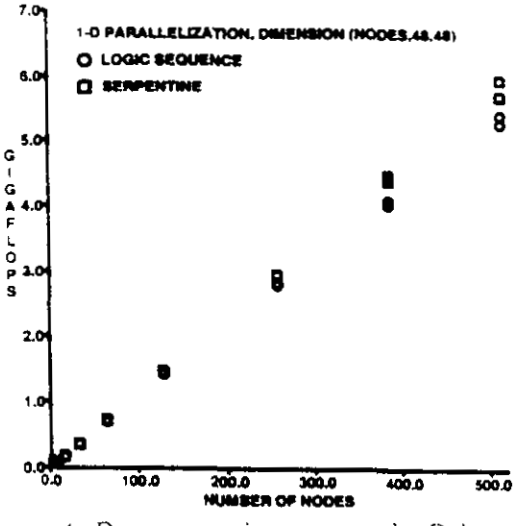

Figure 4: Data processing rate on the Delta

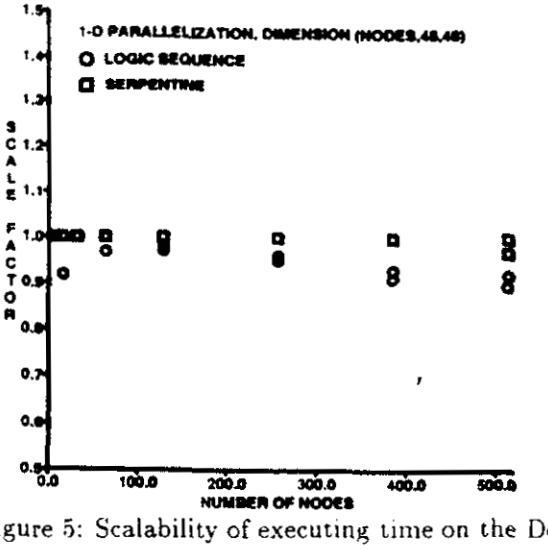

Figure 5: Scalability of executing time on the Delta

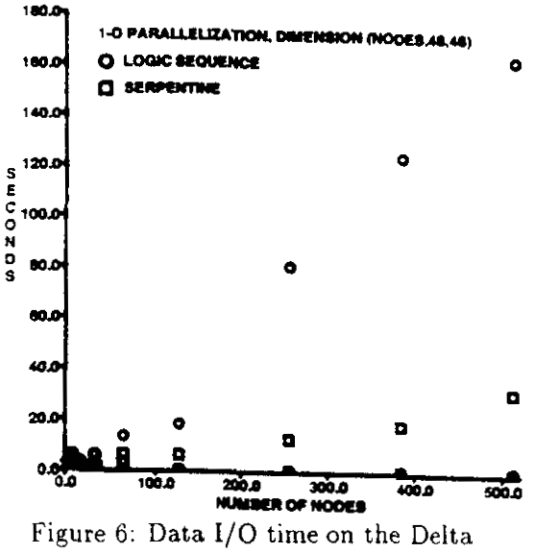

Figure 6: Data I/O time on the Delta

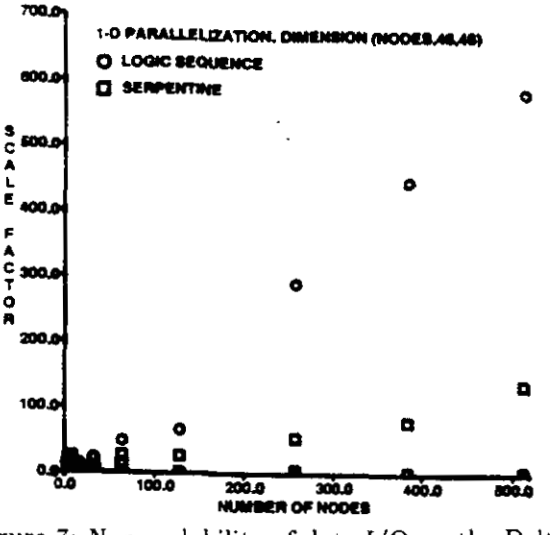

Figure 7: Non-scalability of data I/O on the Delta

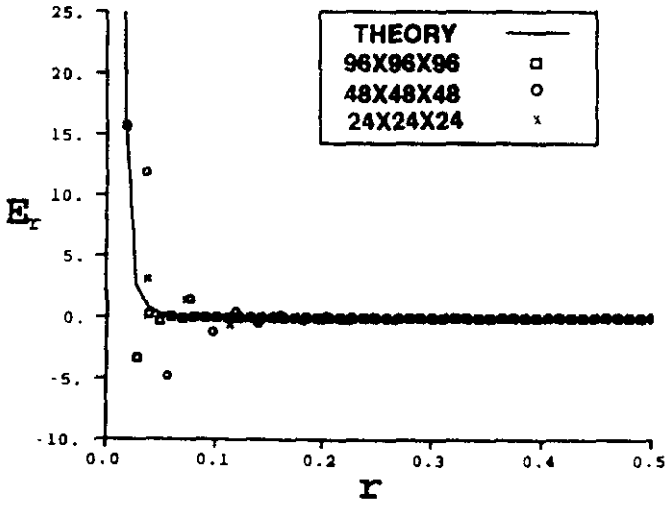

Figure 8: Comparison of computed radial components of the electric fields with theory

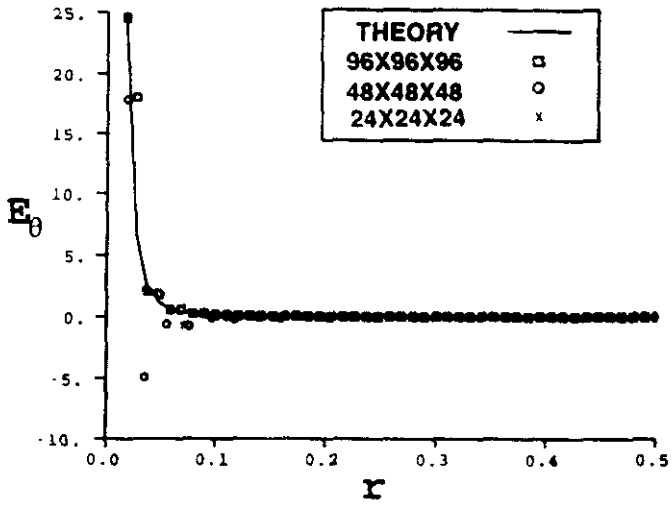

Figure 9: Comparison of computed circumferential components of the electric fields with theory

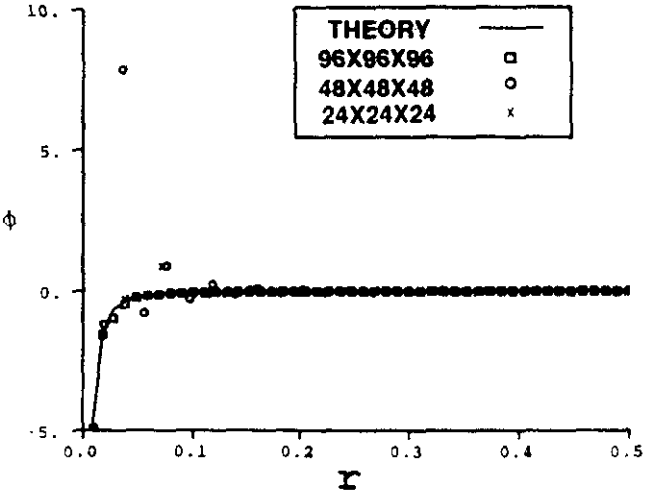

Figure 10: Comparison of computed azimuthal components of the magnetic fields with theory