THE HYPERCUBE IMPLEMENTATION OF THE FINITE-DIFFERENCE TIME-DOMAIN (FD-TD) METHOD FOR ELECTROMAGNETIC WAVE SCATTERING

Daniel Steven Katz
EECS Department, Technological Institute
Northwestern University
Evanston, IL 60208

Presented as the report of
EECS C99 work done within
the EECS Department Honors Program
during the year 1987-88
under Dr. Allen Taflove, Professor

Introduction

The Hypercube (Intel iPSC) is a parallel-processing computer comprised of one master controller and an array of 2^n nodes, where n may be varied from 0 to 4. A finite-difference time-domain (FD-TD) code, which was developed by Kane Yee[1] and further expanded by Allen Taflove[2], can be divided naturally among the nodes, so that the time to solve a problem can be reduced by a factor of approximately 2^n. Of course, the implementation on the Hypercube adds a certain amount of overhead for communication between the nodes which lowers the time reduction afforded by this implementation.

This report discusses two methods which seem reasonable for the implementation of the FD-TD code in one dimension. It then explains the choice between the methods based on benchmark testing, and discusses the implementation of the chosen method in two dimensions, including benchmark times. The report concludes with a discussion of the pros and cons of the Hypercube implementation of the FD-TD method.

---

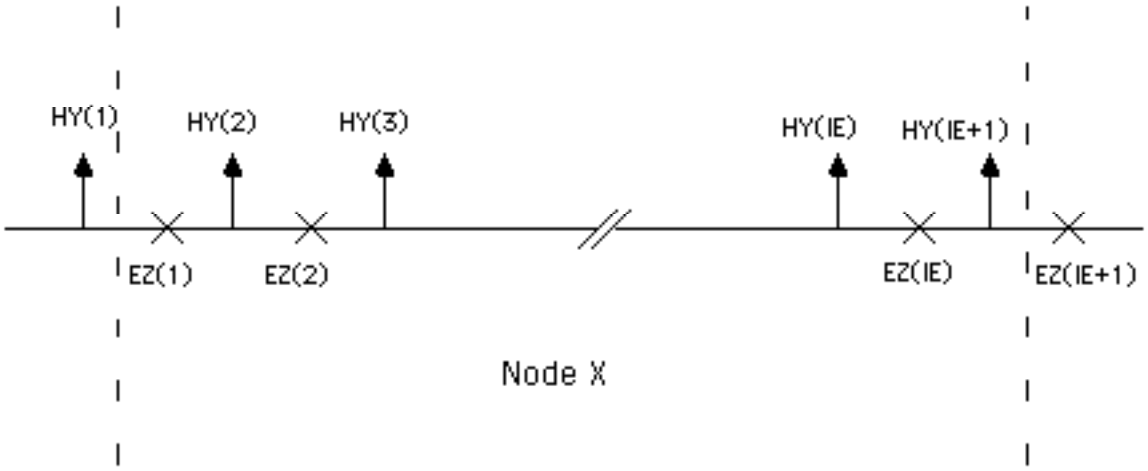
[1] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas Propagation*, vol. AP-14, pp. 302-307, May 1966.

[2] A. Taflove and M. E. Brodwin, "Numerical Solution of steady-state electromagnetic scattering problems using the time dependent Maxwell's equations," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-23, pp. 623-630, Aug. 1975.

One Dimension

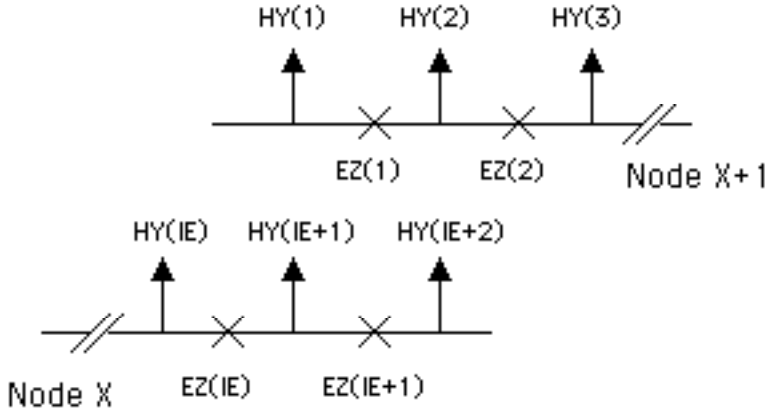
The Two Methods in One Dimension

Method 1



In each time step, node X receives $H_y(1)$ from node X-1,and sends $H_y(IE+1)$ to node X+1. It then calculates electric field components from $E_z(1)$ to $E_z(IE)$. After this, $E_z(IE+1)$ is received from node X+1, and $E_z(1)$ is transmitted to node X-1, and then magnetic field components from $H_y(2)$ to $H_y(IE+2)$ are calculated.

Method 2



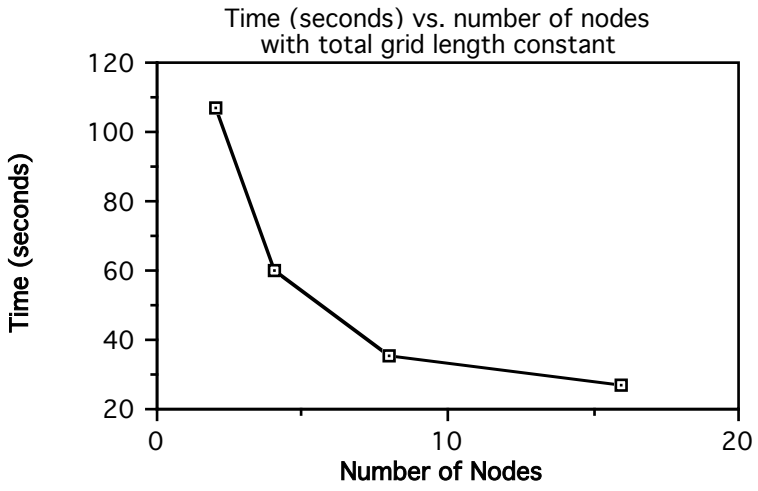
In each time step, node X sends $H_y(IE+1)$ to node X+1 and receives $H_y(IE+2)$ from node X+1. It also sends $H_y(2)$ to node X-1 and receives $H_y(1)$ from node X-1. Node X can then compute $E_z(1)$ to $E_z(IE+1)$, and then $H_y(2)$ to $H_y(IE+1)$. It should be noted that $E_z(IE+1)$ of node X is the same as $E_z(1)$ of node X+1, and that $E_z(1)$ of node X is equivalent to $E_z(IE+1)$ of node X-1.

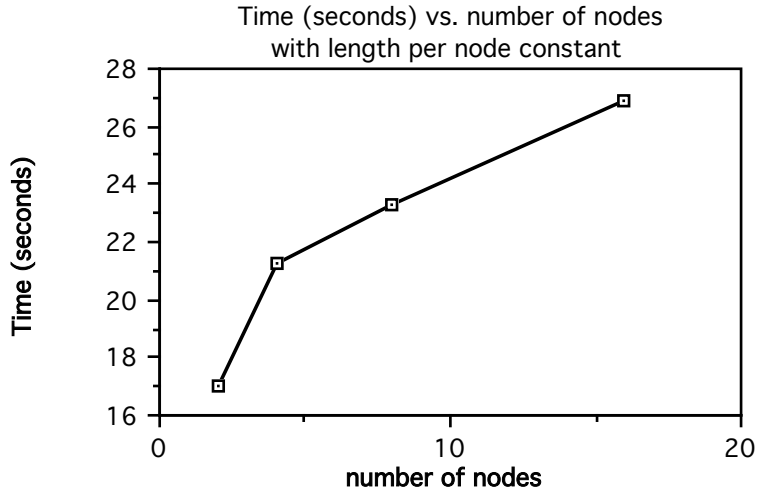
Hypercube Benchmark Times

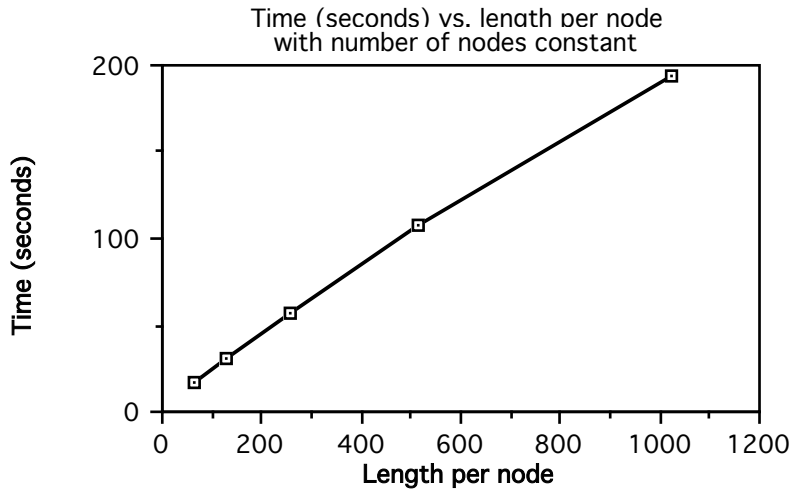
$\Delta X = \frac{\lambda}{20}$, f = 200 MHz, $\Delta T = \frac{\Delta X}{c}$, free space with hard source at left edge and perfectly absorbing RBC

Length (cells)	Number of Nodes	Length per Node	Number of Time Steps	Method #1 Time (seconds)	Method #2 Time (seconds)
1024	2	512	1000	107.04	107.14
1024	4	256	1000	59.76	59.69
1024	8	128	1000	39.69	35.71
1024	16	64	1000	28.33	26.92
128	2	64	1000	16.92	17.02
256	4	64	1000	21.59	21.26
512	8	64	1000	28.09	23.29
1024	16	64	1000	28.33	26.92
2048	2	1024	1000	194.02	194.13
1024	2	512	1000	107.04	107.14
512	2	256	1000	57.24	57.35
256	2	128	1000	30.52	30.63
128	2	64	1000	16.92	17.02

It appears that method #1 is slightly faster for small numbers of nodes, but as the number of nodes increases, method #2 becomes better. For this reason, method #2 is used for the analysis of data which follows, and for the two dimensional implementation.

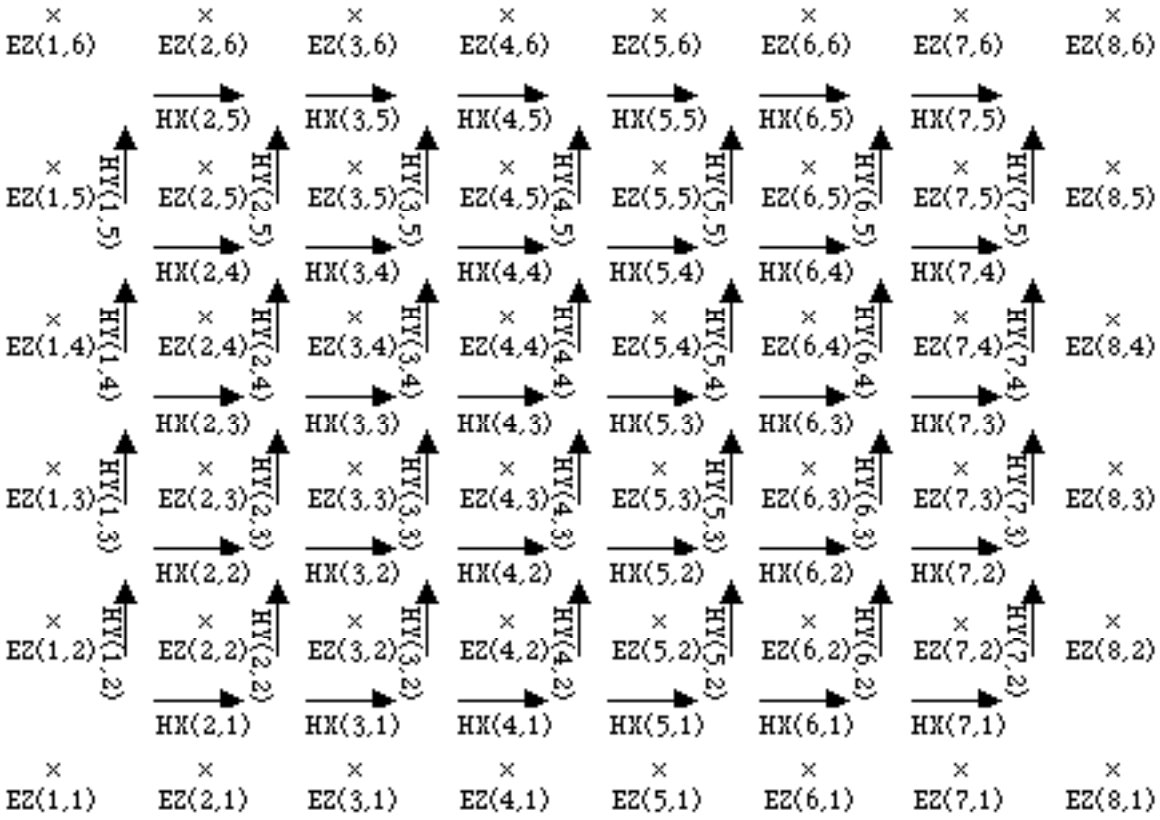

Time (seconds) vs. number of nodes
with total grid length constant
Time (seconds)
120
100
80
60
40
20
0
10
20
Number of Nodes


Time (seconds) vs. number of nodes
with length per node constant
Time (seconds)
28
26
24
22
20
18
16
0
10
20
number of nodes


Time (seconds) vs. length per node
with number of nodes constant
Time (seconds)
200
100
0
0
200
400
600
800
1000
1200
Length per node

Two Dimensions

The Method in Two Dimensions

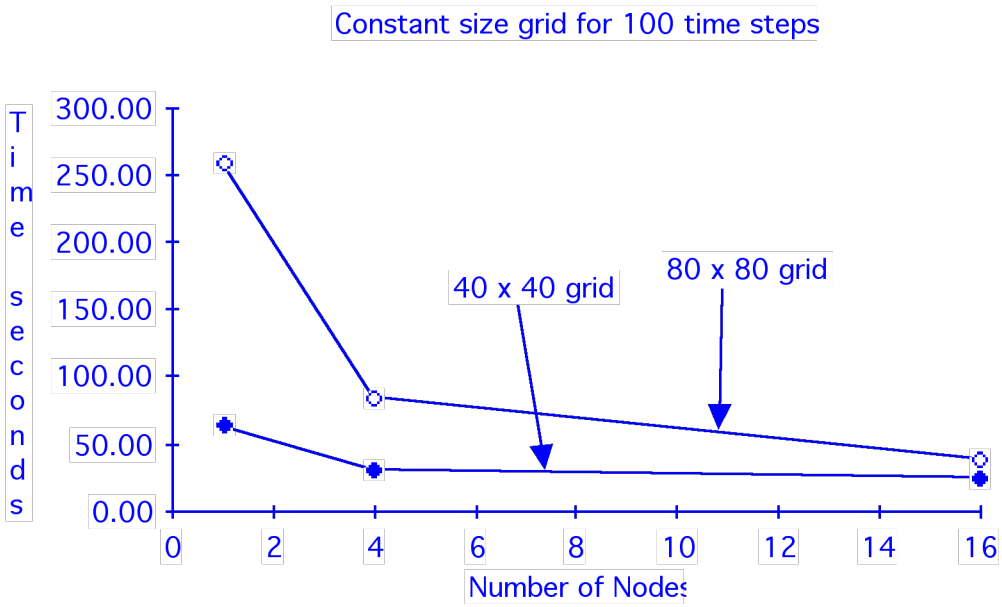


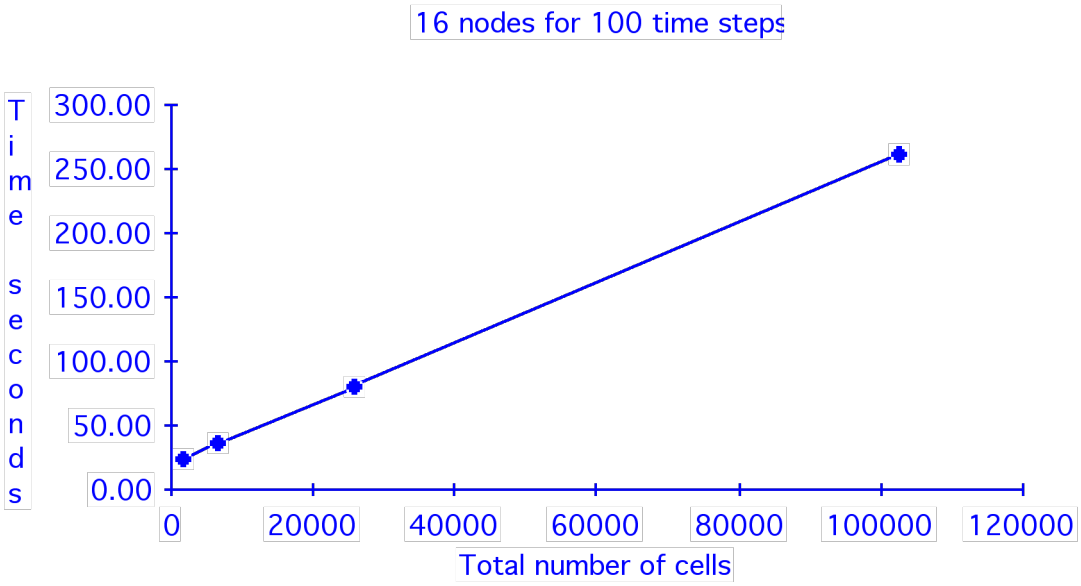
The above diagram illustrates one node in the two dimensional FD-TD model. The node size is 664. All of the H_X and H_Y components may be calculated with the normal FD-TD algorithm, as may the E_Z components which are not on the edges. These edge components are either transmitted from adjacent nodes, in which case the E_Z values in the next row or column will be transmitted to that adjacent node, or they may be calculated using Mur radiation boundary conditions if there is no adjacent node. Thus, the total dimension of an 464 array of nodes in which each node is of dimension 664 would be 26618 (4*6+264*4+2). Total field/scattered field computations may be done by realizing the overall coordinate of each component, as well as its local coordinate within its node.

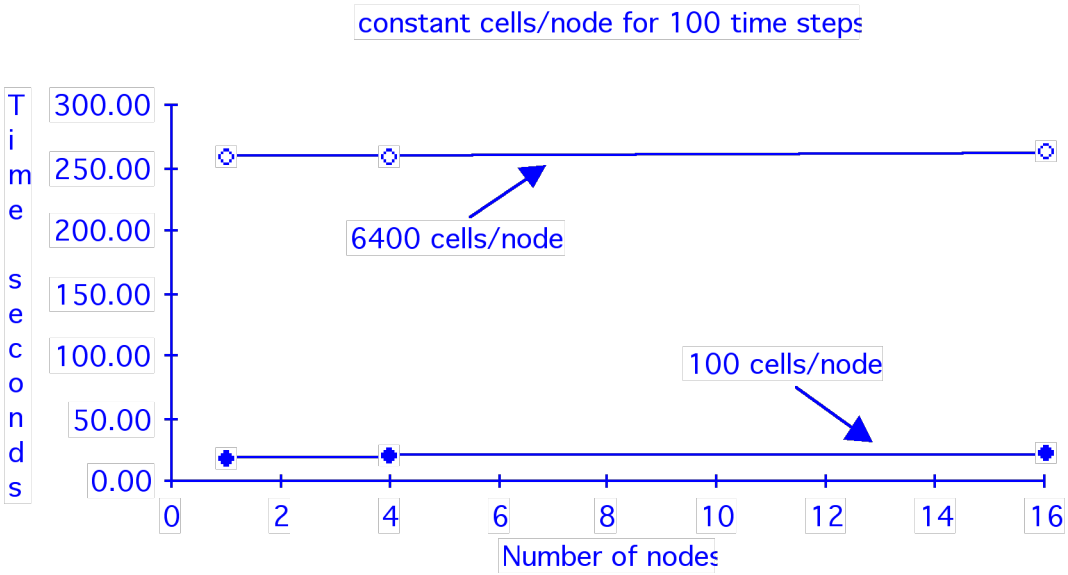
Hypercube Benchmark Times

$\Delta X = \frac{\lambda}{20}$, f = 200 MHz, $\Delta T = \frac{\Delta X}{2c}$, Total field/Scattered field implementation with Mur radiation boundary conditions.

Total Size (cells)	Dimension of Nodes	Size per Node	Time Steps	Number of (seconds)	Time
10610	161	10610	100	19.51	
20620	262	10610	100	21.43	
40640	464	10610	100	23.25	
80680	161	80680	100	259.20	
1606160	262	80680	100	259.65	
3206320	464	80680	100	262.35	
40640	161	40640	100	63.74	
40640	262	20620	100	30.27	
40640	464	10610	100	23.25	
80680	161	80680	100	259.20	
80680	262	40640	100	84.46	
80680	464	20620	100	37.66	
40640	464	10610	100	23.25	
80680	464	20620	100	36.00	
1606160	464	40640	100	81.18	
3206320	464	80680	100	262.35	




16 nodes for 100 time steps
Time seconds
300.00
250.00
200.00
150.00
100.00
50.00
0.00
0
20000
40000
60000
80000
100000
120000
Total number of cells


constant cells/node for 100 time steps
Time seconds
300.00
250.00
200.00
150.00
100.00
50.00
0.00
6400 cells/node
100 cells/node
0
2
4
6
8
10
12
14
16
Number of nodes

Conclusions

It appears that as the number of nodes is increased, a limiting time is approached, which is that of the communication between nodes. A Hypercube of dimension 2^4 or 2^5 would be optimal, in terms of performance versus cost of the computer. However, a 386-based Hypercube would substantially outperform the 286-based Hypercube on which these results were based, and the optimal dimension might increase due to reduced communication time. More importantly, it also has been shown that given a constant number of cells per node, the total grid size may be expanded by using more nodes of the same size without much loss of running time.

The major problem with this implementation is that of inserting the scattering object, since different portions of the scatterer lie in each node, and there is some overlap between nodes. This is simply a problem of memory, though, and as long as there is sufficient memory, it can be ignored. Another problem is that the code is more difficult to understand for one who is not an expert in Fortran, but once the code has been written, the only modification which must be made is to the scatterer specification section.

It appears that, in general, the Hypercube implementation is better and faster than the standard personal computer implementation.