PARALLELISATION OF A MULTI-GRID FDTD ELECTROMAGNETIC APPLICATION CODE FOR DISTRIBUTED MEMORY SYSTEMS

P Chow [1], T Kubota [2], S Johnson [3], P Leggett [3], C Ierotheou [3]

[1] Fujitsu Laboratories of Europe Limited, UK

[2] Fujitsu Limited, Japan

[3] Parallel Software Products Inc., USA

SUMMARY

In this paper we report on the parallelisation of a multi-grid finite-difference time-domain (MG-FDTD) electromagnetic code that includes functions for multi-level local grid embedding in regions of interest. A commercial computer aided parallelisation tool was used to help with 85 per cent of the parallelisation, significantly reducing the time required for the task. On a PC cluster system of 16 nodes, a mean speedup of over 14 was achieved for the FDTD part with no variable field output, while for the multi-grid FDTD the speedup is between 8.9 and 13.5 dependent on model.

PARALLELISATION OF A MULTI-GRID FDTD ELECTROMAGNETIC APPLICATION CODE FOR DISTRIBUTED MEMORY SYSTEMS

1: Introduction

Manual code parallelisation is often time consuming and prone to errors. It is generally considered to be more difficult to parallelise applications for distributed memory systems than for shared memory computers. Issues such as data locality, communication between processing elements, input and output all have to be addressed for efficient parallelisation on distributed memory systems. But systems such as commodity PC clusters are now so affordable that it is worthwhile to parallelise codes for these platforms. Using appropriate computer aided parallelisation tools can significantly reduce the time required and eliminate human coding errors in the process. In this paper we describe the full parallelisation of a multi-grid finite-difference time-domain (FDTD) electromagnetic industrial application code using a computer aided parallelisation tool called ParaWise.

In the sections that follow, we first describe of the electromagnetic code and the parallel strategy used. Parallel results are then given and finally we present our conclusions.

2: Electromagnetic code

In this section the application code is described. The electromagnetic code is based on the finite-difference time-domain (FDTD) method [1, 2] and uses the multi-grid technique for embedding fine grids in regions of interest. First the multi-grid FDTD method is introduced, next the block-solve enhancement, and lastly the multi-level element is discussed.

2.1: Multi-grid FDTD

The multi-grid FDTD method [3–9] solves Maxwell’s equations based on the finite-difference time-domain (FDTD) method for the calculation of electromagnetic wave propagation by strategically embedding finer grids in regions that require high resolution (rather than using fine grid everywhere). Figure 1 illustrates the grids of normal FDTD and multi-grid FDTD methods. The multi-grid approach greatly reduces memory and computational cost compared with an FDTD method of similar accuracy. Furthermore, the Cartesian grid used in the finite-difference scheme is considered the most efficient in terms of memory requirement and computational efficiency.

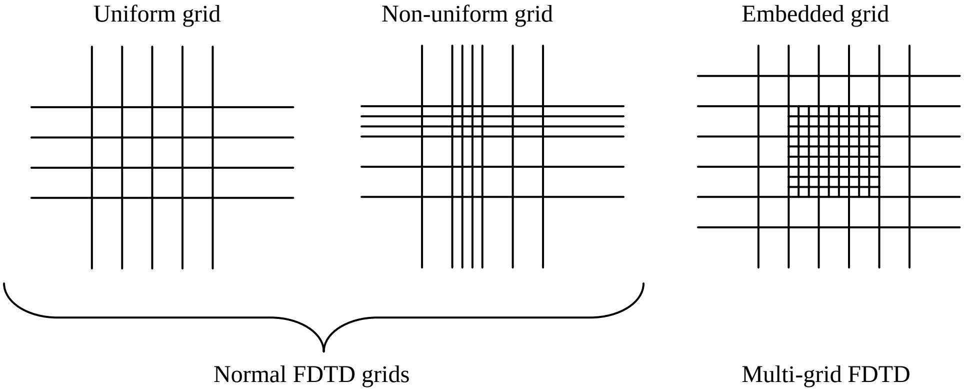

Figure 1: Comparison of grid types used in FDTD and multi-grid FDTD methods

2.2: Block-solve multi-grid FDTD

For most calculations the multi-grid FDTD method is fast and efficient. However, there are situations where the embedded Cartesian grid is very inefficient, such as elongated irregular signal-line type structures. Owing to the rectangular nature of Cartesian grids, they can cover large empty spaces. Figure 2 shows such a model where the embedded grid (the rectangle covering the signal-line structure) is covering some 50 per cent of empty signal-line space as illustrated by the two dotted-line ovals. Naturally, removing such empty space will reduce memory requirements and the number of calculations thus shortening the solution time. Figure 3 shows a configuration of three embedded grids covering the signal-line structure.

Applying the multi-grid FDTD method directly to the model in Figure 3 will give an incorrect solution. This is because the temporal space at the adjoining interface between the grids in the fine-grid time-step level is not directly communicated. Instead, it is via the coarse grid where it is interpolated with the rest of the boundary from coarse-grid values. Therefore, a modification to the solution procedure to include the exchange of values between adjoining grids at each embedded time-step level is required. The block-solve solution procedure for multi-grid FDTD is detailed in [10] and will not be repeated here.

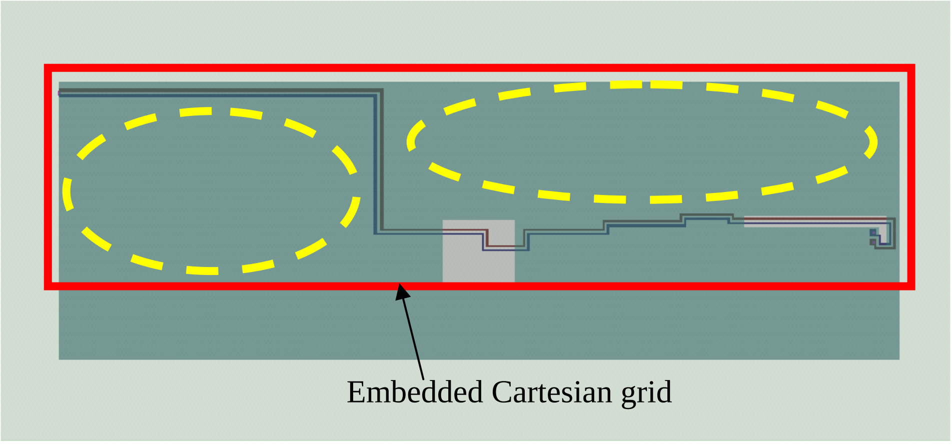

Figure 2: **Embedded grid covering an irregular signal-line structure.**

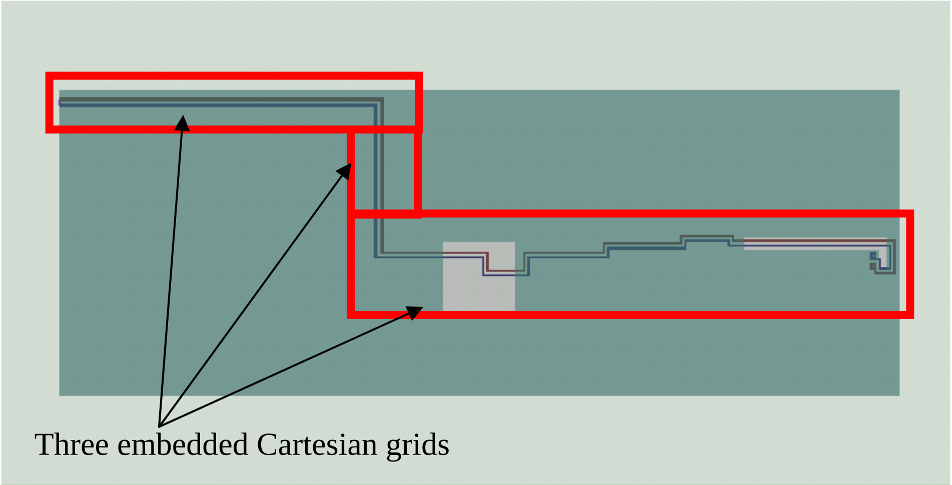

Figure 3: **Multiple Cartesian grids covering the signal-line structure.**

2.3: Block-solve multi-level multi-grid FDTD

Figure 4 shows a multi-level grid configuration (in two dimensions) of three levels of grid embedding, each embedded grid finer than the one before. The multi-level multi-grid FDTD method [11] provides a refinement process where the grid cell aspect scale changes gradually as opposed to a severe change. This refinement applies to both spatial and temporal spaces. Thus, a gradual refinement is commonly a more stable approach than a severe change all at once.

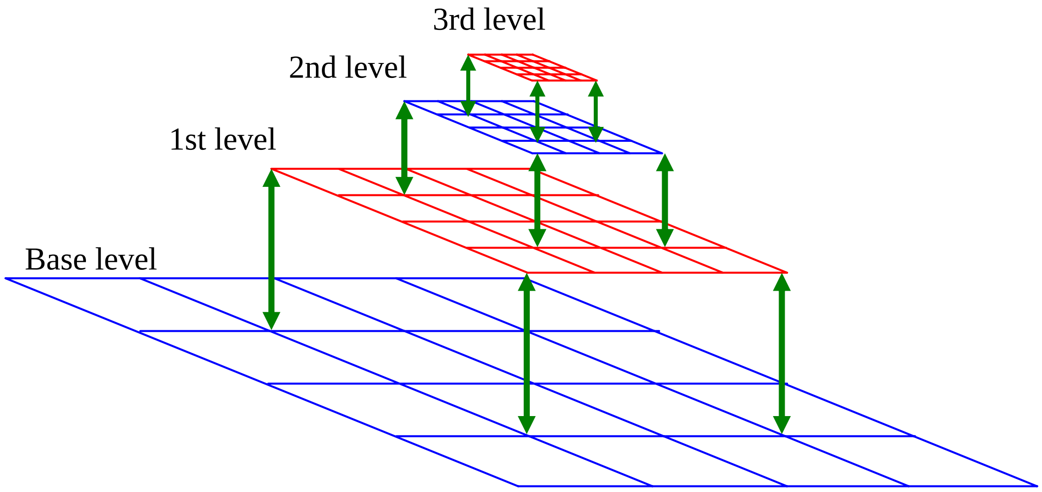

Figure 4: Multi-level grid configuration.

3: Parallelisation Strategy

The strategy for parallelisation here is first to parallelise the FDTD algorithm and the input/output elements, this representing about 85 per cent of the electromagnetic code. This is followed by the multi-grid and multi-level elements, and finally consideration is given to the load balancing. How to efficiently parallelise the FDTD method is described in a textbook by Wenhua Yu et al. [12]. The approach needs only nearest neighbour communications for each of the field variables, electric and magnetic. The domain partition is a simple equal division of grid points in each dimension.

3.1: FDTD, input and output

Instead of a manual parallelisation of the FDTD procedure, representing some 85 per cent of the electromagnetic code, we used a commercial computer aided parallelisation tool called ParaWise [13] from Parallel Software Products Inc. to perform the parallelisation. This significantly reduces the time required and eliminates human coding errors in the process compared with the manual approach. For shared memory systems the output parallel source code has OpenMP directives, whereas for distributed memory systems MPI communication calls are inserted. At present ParaWise uses a master node model for input and output to achieve file compatibility. This means for distributed memory systems input is read from the master node and distributed to all other nodes. For output, the reverse is applied, with all output data from the other nodes being communicated to the master to perform the output. This is fine for input that is done once at the start and output that is done once at the end of the simulation. For interval output such as for animations, such an output model is inefficient and can lead to deterioration in performance. It is more efficient for each node to output to a temporary file and to combine these together at the end to achieve file compatibility. Naturally, when any downstream programs such as visualisation can read the parallel temporary file then the combination step is not required.

3.2: Multi-grid, block-solve and multi-level

The computer aided tool is not able to handle the multi-grid, block-solve and multi-level elements because of the grid embedding, and thus this has to be done manually. Fortunately, the same communications arrangement for the FDTD can be applied and greatly simplifies the parallelisation. It means the communication space is a subset of the coarse grid domain partition. Figure 5 shows a coarse grid with nine domain partitions with two embedded grids (in red). The coarse grid domain partition subset for the small embedded grid comprises nodes 1, 2, 4 and 5, while for the large embedded grid the nodes are 4, 5, 6, 7, 8 and 9. This is a simple change of the domain partition list before performing each embedded grid and on finishing, return the domain partition list back to its previous state. This mechanism is naturally applicable for the multi-level element. Apart from the array indexing changes required for domain partition boundaries, no other modifications are required.

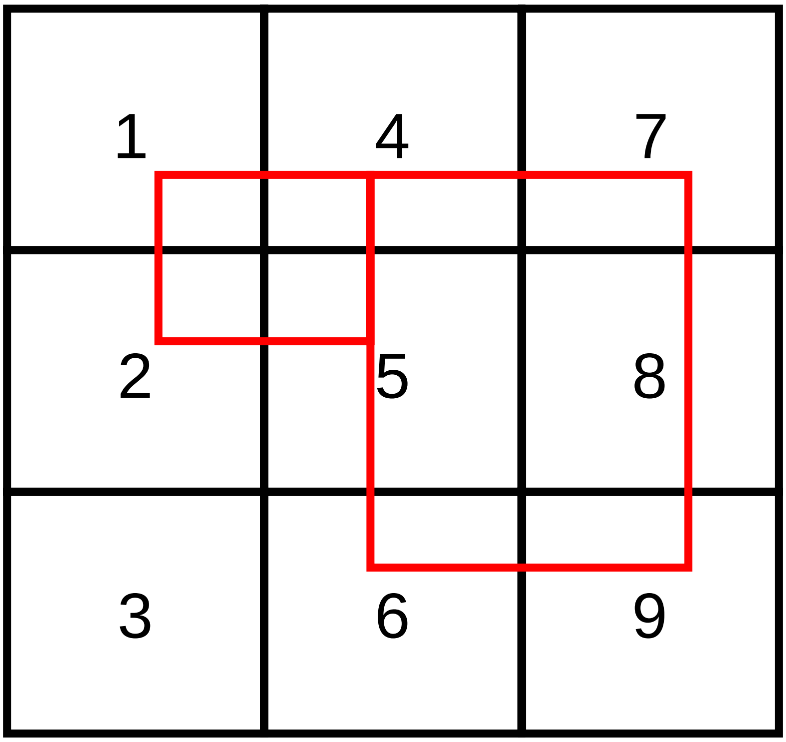

Figure 5: **Domain partition for multi-grid and block-solve.**

3.3: Load balancing

Using the domain partition calculated for the coarse grid will in general not lead to acceptable load balancing for the example in Figure 5. This is because nodes 4 and 5 have the highest work load (involved in two embedded grids) and node 3 the lowest work load (involved in no embedded grid). The rest of the nodes are all involved in one embedded grid.

A more load balanced approach is to include the embedded grid information into the calculation to give a coarse grid domain partitioning that represents a more even the work load.

4: Parallel results

The following results are obtained on a PC cluster based systems with a total of 16 nodes. Each node has a 2.4 GHz Xeon processor with 2GB memory. The cluster interconnect is Myrinet 2000 using Myricom's MPICH communication library.

4.1: FDTD results

Table 1 shows the result of a model with grid size of 301 x 301 x 301 that ran for 1000 time steps with no variable field output. The speedup performance for 16 nodes is 14.43 (based on total time), whereas on two and four nodes the total time speedups are 2.01 and 3.96 respectively.

Nodes	Solution Time (s)	Solution Speedup	Total Time (s) merging files, etc.	Total Time Speedup
serial	2317.42			
2 (1 x 2)	1145.68	2.02	1151.45	2.01
4 (4 x 4)	581.58	3.99	584.72	3.96
9 (3 x 3)	287.60	8.06	289.13	8.02
16 (4 x 4)	159.56	14.52	159.56	14.43

Table 1: Time results of FDTD model with no variable field output.

Table 2 shows the result of a model with grid size of 134 x 134 x 134 that ran for 3584 time steps with electric, magnetic and SAR fields output. The speedup performance for interval output of 16 nodes is 7.79 based on total time, whereas on two and four nodes the total time speedups are 1.88 and 3.33 respectively.

Nodes	Solution Time (s)	Solution Speedup	Total Time (s) merging files, etc.	Total Time Speedup
serial	778.47			
2 (1 x 2)	377.15	1.98	397.15	1.88
4 (4 x 4)	203.73	3.67	224.54	3.33
9 (3 x 3)	123.60	6.30	146.65	5.31
16 (4 x 4)	76.85	10.13	99.90	7.79

Table 2: Time results with electric, magnetic and SAR field output.

4.2: Multi-grid FDTD results

Table 3 shows the result of a model with grid size of 288 x 216 x 143 that ran for 7,795,863 time steps with no fields output. The speedups are 14.04, 13.57, 12.70 and 8.93 respectively for fine grid, multi-grid, block multi-grid (using three blocks), and block multi-grid (using four blocks). If we compare the fastest time with the serial fine grid time, a speedup of over 119 is achieved.

Grid Model	Serial Time (days)	Parallel Time (days) 16 nodes (2 x 8)	Speedup
Fine grid	67.11	4.78	14.04
Multi-grid	14.38	1.06	13.57
Block MG (3 blocks)	8.51	0.67	12.70
Block MG (4 blocks)	5.00	0.56	8.93

Table 2: Time results of multi-grid FDTD with no variable field output.

5: Conclusions

For the parallelisation of an electromagnetic code on a PC cluster system of 16 nodes, a mean speedup of over 14 was achieved for the FDTD part with no variable field output. For variable field output the speedup is 7.79, with file merging taking 30 percent of the total time.

In multi-grid FDTD the speedup is between 13.5 and 8.9 dependent on model with no variable field output. If the parallel block multi-grid time is compared with the serial fine grid time, a speedup of over 119 is achieved.

REFERENCES

1. YEE, KS - Numerical Solution of Initial Boundary Value Problems Involving Maxwell´s Equations in Isotropic Media, IEEE Trans. Antennas Propagation, Vol 14, pp. 302–307, 1996
2. TAFLOVE, A – Computational Electrodynamics: The Finite-Difference Time-Domain Method, Artech House, Boston, 1995
3. KIM, I S, HOEFER, W J R – A Local Mesh Refinement Algorithm for the Time Domain Finite Difference Method Using Maxwell’s Curl Equations, IEEE Trans. Microwave Theory Tech., Vol 38, pp. 812–815, 1990
4. ZIVANOVIC, S S, YEE, K S, MEI, K K – A Subgridding Method for the Time-Domain Finite-Difference Method to Solve Maxwell’s Equations, IEEE Trans. Microwave Theory Tech., Vol 39, pp. 471–479, 1991
5. MONK, P – Sub-Gridding FDTD Schemes, J. Applied Computational Electromagnetic Society, Vol 11, pp. 37–46, 1996
6. THOMAS, P, WEILAND, T – A Consistent Subgridding Scheme for the Finite Difference Time Domain Method, Int. J. Numerical Modelling: Electronic Networks, Devices & Fields, Vol 9, pp. 359–374, 1996
7. CHEVALIER, M W, LUEBBERS, R J, CABLE, V P – FDTD Local Grid with Material Traverse, IEEE Trans. Antennas Propagation, Vol 45, pp. 411–421, 1997
8. OKONIEWSKI, M, OKONIEWSKA, E, STUCHLY, M A – Three-Dimensional Subgridding Algorithm for FDTD, IEEE Trans. Antennas Propagation, Vol 45, pp. 422–429, 1997
9. WHITE, M J, YUN, Z, ISKANDER, M F – A New 3D FDTD Multigrid Technique with Dielectric Traverse Capabilities, IEEE Trans. Microwave Theory Tech., Vol 49, pp. 422–430, 2001
10. CHOW, P, KUBOTA, T, NAMIKI, T – A Block-Solve Multigrid-FDTD Method, The 22nd International Review of Progress in Applied Computational Electromagnetics, (ACES 2006) conference in March 2006, Miami, Florida, USA
11. CHAILLOU, S, WIART, J, TABBARA, W – A Subgridding Scheme Based on Mesh Nesting for FDTD Method, Microwave and Optical Technology Letters, Vol 22, pp. 211–214, 1999
12. YU, MITTRA, SU, LIU and YANG – Parallel Finite-Difference Time-Domain Method, Artech House, Boston, 2006
13. ParaWise, Parallel Software Products Inc., USA (www.parallelsp.com)