

PARALLEL FDTD SIMULATION USING NUMA ACCELERATION TECHNIQUE

X.-M. Guo^{1,*}, Q.-X. Guo¹, W. Zhao², and W.-H. Yu³

¹School of Information Engineering, Communication University of China, Beijing 100024, China

²School of Maths and Information Engineering, Hebei Normal University of Science and Technology, Hebei 066000, China

³Electromagnetic Communication Laboratory, Pennsylvania State University, University Park, PA 16802, USA

Abstract—In this paper, we introduce a new non-uniform memory access (NUMA) acceleration algorithm for parallel finite-difference time-domain (FDTD) method on NUMA architecture workstation. We compare the performance of parallel FDTD method with and without the NUMA acceleration technique. An ideal test case and an engineering example show that the NUMA acceleration technique can efficiently improve the computing performance of FDTD parallel applications.

1. INTRODUCTION

The finite-difference time-domain (FDTD) method was originally developed by Yee [1], in 1966. Because of its simplicity and flexibility, this method has become one of the popular algorithms in solving a wide variety of problems in electromagnetics. The nature of the FDTD [2–4] method is that simulation of big and complicated electromagnetic (EM) field problems requires a vast amount of computer operational memory and runtime. Parallel-processing techniques [5–7] have been broadly applied in FDTD method, and the parallel-processing FDTD further accelerates the FDTD simulation by distributing the job to multiple processors.

Today, both Intel and AMD use the non-uniform memory access (NUMA) architecture in the multiple CPU chipset to achieve the best

Received 17 October 2011, Accepted 18 November 2011, Scheduled 28 November 2011

* Corresponding author: Xiao-Mei Guo (gxmqin@163.com).

performance at a relatively low price. In this paper, we present a NUMA acceleration algorithm in parallel FDTD methods. The results show that NUMA acceleration technique can efficiently improve the computing performance of FDTD parallel applications.

2. FDTD METHOD

The Yee algorithm using the coupled Maxwell's curl equation to solve both electric and magnetic fields in time and space.

The Maxwell equations describe how a time varying electric field E generates a time varying magnetic field H and vice versa. In a region of space without electric and magnetic current sources, the Maxwell curl equations are:

$$\nabla \times \mathbf{H} = \varepsilon \frac{\partial \mathbf{E}}{\partial t} + \sigma \mathbf{E} \quad (1)$$

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} - \sigma_m \mathbf{H} \quad (2)$$

where ε and μ are constants called the electric permittivity and the magnetic permeability respectively, σ and σ_m are the electric conductivity and the equivalent magnetic loss.

It is well known that, in Yee's algorithm, the electric and magnetic fields are defined on an intertwined double mesh, where electric field components are circulated by four magnetic field components and magnetic field components are circulated by four electric field components. For example, the electric field E_x can be expressed as follows [8]:

$$\begin{aligned} E_x^{n+1} \left(i + \frac{1}{2}, j, k \right) = & CA \left(i + \frac{1}{2}, j, k \right) \cdot E_x^n \left(i + \frac{1}{2}, j, k \right) \\ & + CB \left(i + \frac{1}{2}, j, k \right) \cdot \left[H_z^{n+\frac{1}{2}} \left(i + \frac{1}{2}, j + \frac{1}{2}, k \right) \right. \\ & - H_z^{n+\frac{1}{2}} \left(i + \frac{1}{2}, j - \frac{1}{2}, k \right) + H_y^{n+\frac{1}{2}} \left(i + \frac{1}{2}, j, k - \frac{1}{2} \right) \\ & \left. - H_y^{n+\frac{1}{2}} \left(i + \frac{1}{2}, j, k + \frac{1}{2} \right) \right] \end{aligned} \quad (3)$$

where

$$\begin{aligned} CA(i, j, k) &= \frac{1 - \frac{\sigma(i, j, k) \Delta t}{2\varepsilon(i, j, k)}}{1 + \frac{\sigma(i, j, k) \Delta t}{2\varepsilon(i, j, k)}} \\ CB(i, j, k) &= \frac{\Delta t}{\varepsilon(i, j, k) \Delta s} \cdot \frac{1}{1 + \frac{\sigma(i, j, k) \Delta t}{2\varepsilon(i, j, k)}} \end{aligned}$$

As we can see from (3), the iterations of each component are only related with four components around it without consideration of the whole field. Parallelism is an inherent property of FDTD, so that computation time should be reduced significantly using the parallel architecture.

3. NUMA ACCELERATION TECHNIQUE

First, to make comparisons, we present a MPI-OPENMP hybrid parallel FDTD algorithm:

- MPI initialization (MPI-Init (int*argc, char**argv));
- Using OpenMP multithreading, initialize fields, apply initial conditions;
- Start time iterations:
 - Using OpenMP multithreading, update the H -field components;
 - Using MPI message passing, exchange the H -field of boundaries;
 - Using OpenMP multithreading, update the E -field components;
 - Apply boundary conditions;
 - Until predetermined time steps is reached;
- End (MPI-Finalize());).

The traditional model for multiprocessor support is Symmetric Multi-Processor (SMP). In this model, each processor has equal access to memory and I/O. As more processors are added, the processor bus becomes a limitation for system performance.

System designers are using NUMA to increase processor speed without increasing the load on the processor bus in recent years. The architecture is non-uniform because each processor is close to some parts of memory and farther from other parts of memory. The processor quickly gains access to the memory to which it is close, while it can take longer to gain access to memory that is farther away.

In NUMA architecture, CPUs are regularly arranged in a smaller box that is so-called compute node. Each node has its own processors and memory, and is connected to a larger cluster system through a cache-coherent interconnect bus [9]. We improve the simulation performance by explicitly pinning a thread to a specified core and data to a specified node. The threads scheduled on processors are located in the same node as the memory being used. If a thread is running and accessing data on the same node, it is considered as a local access. If a thread is running on one node but accessing data resident on a different node, it is considered as a remote access. Accessing data remotely is slower than accessing data locally [9]. If we do not bind a thread to a particular core, the thread often accesses data remotely [10, 11], but if

we bind a thread to a particular core, the thread is running on this core; the memory allocation to the thread is in the same node; the thread cannot access other memory remotely. So the method can reduce the runtime of the applications.

The algorithms are described as follows:

- MPI initialization (MPI-Init (int*argc, char**argv));
- Using OpenMP multithreading, set the number of threads;
- Using GetNumaHighestNodeNumber (&HighestNodeNumber), Retrieve the highest numbered node in the system;
- Using GetNumaNodeProcessorMask (nodenumber, &ProcessorMask), Retrieve the processor mask for the specified node;
- Using GetCurrentProcess function, Retrieve the current process;
- For processId = 0 to n do
 Using VirtualAllocExNuma function, Reserve or commit a region of memory within the virtual address space of the specified process, and specify the NUMA node for the physical memory;
 Using SetProcessAffinityMask (GetCurrentProcess(), ProcessAffinityMask), bind a thread on a particular core;
 End for;
- Start time iterations:
 Using OpenMP multithreading, update the H -field components;
 Using MPI message passing, exchange the H -field of boundaries;
 Using OpenMP multithreading, update the E -field components;
 Apply boundary conditions;
 Until predetermined time steps is reached;
- End (MPI-Finalize());

Now, we use the parallel FDTD code to simulate an ideal test case that is a hollow box with the simplest excitation and output, and its domain is truncated by using the perfect electric conductor (PEC). Excitation point source is Gaussian pulse. On a 4-CPU NUMA architecture workstation, we ran the problem with different sizes using parallel FDTD code without NUMA acceleration, and the parallel FDTD code with NUMA acceleration. We can use the following mathematical analysis to evaluate the algorithms:

$$\text{Performance (Mcells/s)} = \frac{(N_x \times N_y \times N_z) \times \text{Number_of_timesteps}}{\text{Simulation_time (second)}}$$

Figure 1 shows the electric fields with and without NUMA acceleration. The simulation results are in a very good agreement. According to the above mathematical analysis, we have the simulation summary in Figure 2. It is observed from Figure 2 that the NUMA

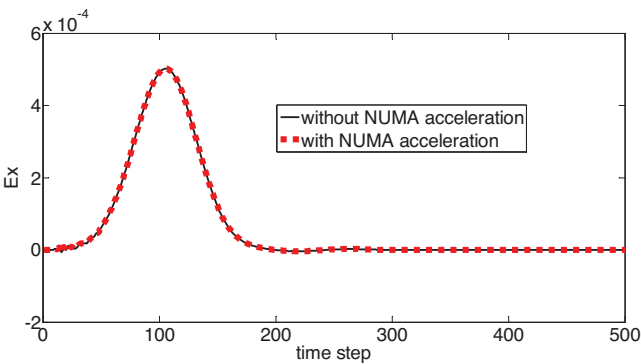


Figure 1. Electric field E_x with and without NUMA acceleration.

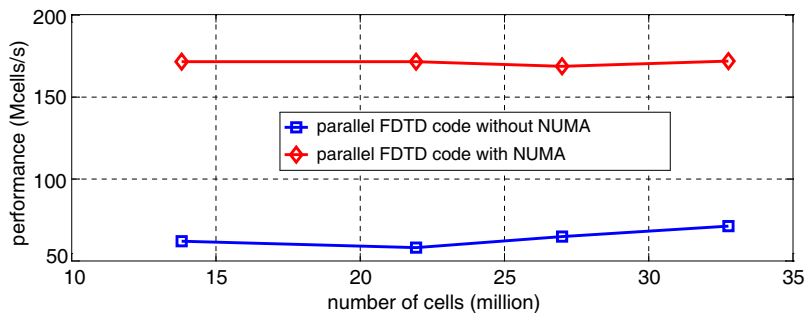


Figure 2. FDTD performance on CPU without NUMA and CPU with NUMA for the ideal test case.

acceleration technique accelerates the FDTD code 2 to 3 times, and the peak computation time of the NUMA FDTD code is 3 times faster than the FDTD code without NUMA.

4. ENGINEERING EXAMPLE

In order to verify the acceleration effect, we now use the parallel FDTD code with and without NUMA acceleration to simulate a practical problem on a 4-CPU workstation (AMD Opteron 6168 1.9GHz processor, 64G RAM) and compare their running time. The actual acceleration factor of the technique depends on the problem type and output options. In our test, we have run many engineering applications on several different NUMA architecture workstations. For most engineering applications, the acceleration factor is between 1.5

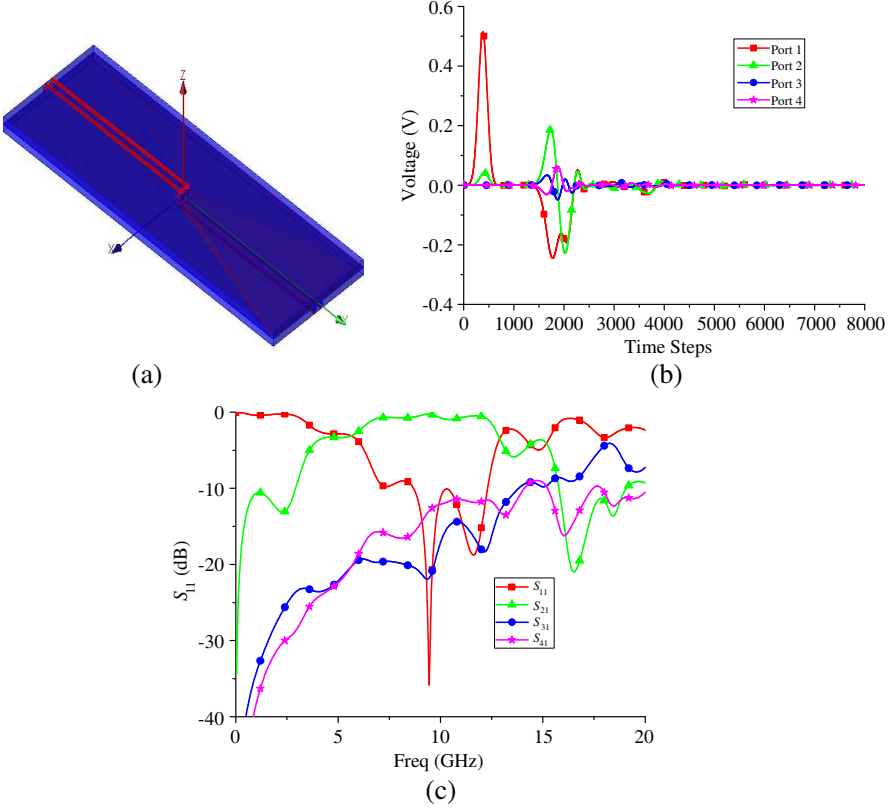


Figure 3. Pair microstrip lines. (a) Configuration. (b) Time domain signal of each port. (c) S -parameter.

and 2.5. Simulation time includes the project preprocessing (including material and mesh generations), iterations and postprocessing.

For example, the configuration of two vias that connect a pair of microstrip transmission lines on the top layer of a three-layer printed circuit board (PCB) with another pair of microstrip transmission lines on the bottom layer of the PCB is simulated. Reflected and transmitted time domain signals and S -parameter are calculated. Configuration and results are shown in Figures 3(a) and (b). Internal plane between two substrates with 0.5 mm thickness is the ground plane, and the thickness of both ground plane and microstrips is 18 μm . The diameter of two vias is 0.5 mm, and the diameters of holes are 0.25 mm. The width of the two microstrips is 0.2 mm, and the gap between the strip lines is 0.7 mm. The total dimensions of the

microwave circuit in the horizontal directions are $30\text{ mm} \times 10\text{ mm}$. It takes 20 minutes 51 seconds to finish 8000 time steps with NUMA and 48 minutes 10 seconds without NUMA. The results with and without NUMA acceleration are the same, shown in Figure 3(c), but the simulation time with the NUMA acceleration has been decreased significantly.

5. CONCLUSION

A NUMA acceleration technique in parallel FDTD methods has been presented in this letter. The technique has achieved an excellent performance. Our examples show that the computation time of the FDTD application with NUMA acceleration technique is decreased significantly.

REFERENCES

1. Yee, K. S., "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, Vol. 14, No. 5, 302–307, May 1966.
2. Garcia, S. G., F. Costen, M. Fernandez Pantoja, L. D. Angulo, and J. Alvarez, "Efficient excitation of waveguides in Crank-Nicolson FDTD," *Progress In Electromagnetics Research Letters*, Vol. 17, 27–38, 2010.
3. Cao, D.-A. and Q.-X. Chu, "FDTD analysis of chiral discontinuities in waveguides," *Progress In Electromagnetics Research Letters*, Vol. 20, 19–26, 2011.
4. Ai, X., Y. Han, C. Y. Li, and X.-W. Shi, "Analysis of dispersion relation of piecewise linear recursive convolution FDTD method for space-varying plasma," *Progress In Electromagnetics Research Letters*, Vol. 22, 83–93, 2011.
5. Jensen, M. A., A. Fijany, and Y. Rahmat-Samii, "Time-parallel computational strategy for FDTD solution of Maxwell's equations," *IEEE Antennas and Propagation Society International Symp.*, Vol. 1, 380–383, Jun. 1994.
6. Guiffaut, C. and K. Mahdjoubi, "A parallel FDTD algorithm using the MPI library," *IEEE Antennas and Propagation Magazine*, Vol. 43, No. 2, 94–103, Apr. 2001.
7. Yu, W., R. Mittra, T. Su, Y. Liu, and X. Yang, *Parallel Finite Difference Time Domain Method*, Artech House, MA, Jun. 2006.

8. Taflove, A. and S. Hagness, *Computational Electromagnetics: The Finite-difference Time-domain Method*, 3rd edition, Artech House, MA, 2005.
9. Jagasia, H., “Performance guidelines for developers on AMD AthlonTM 64 and OpteronTM ccNUMA multiprocessor systems running Microsoft Windows,” Advanced Micro. Devices, May 2006.
10. Broquedis, F., N. Furmento, B. Goglin, R. Namyst, and P.-A. Wacrenier, “Dynamic task and data placement over NUMA architectures: An OpenMP runtime perspective,” *5th International Workshop on OpenMP, IWOMP 2009, Ser. Lecture Notes in Computer Science*, Vol. 5568, 79–92, Springer, Dresden, Germany, Jun. 2009.
11. McCurdy, C. and J. S. Vetter, “Memphis: Finding and fixing NUMA-related performance problems on multi-core platforms,” *Proceedings of ISPASS 2010*, 87–96, 2010.