Performance Improvement of Three-Dimensional Tiled FDTD Kernel Based on Automatic Parameter Tuning

Takeshi Minami[1], Motoharu Hibino[1], Tasuku Hiraishi[2], Takeshi Iwashita[2], Hiroshi Nakashima[2]

[1] Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
[2] Academic Center for Computing and Media Studies, Kyoto University, Kyoto 606-8501, Japan

This paper introduces an automatic tuning method of the tiling parameters required in the implementation of the three-dimensional FDTD method based on time-space tiling. The tuned tiled FDTD kernel was multi-threaded and its performance was evaluated on a multi-core processor. Compared with a naïvely implemented kernel, this tuned FDTD kernel performed better by more than a factor of two.

***Index Terms*— Auto-tuning, Cache, FDTD method, Multithreading, Stencil computation, Time-space tiling**

I. Introduction

The three-dimensional (3-d) FDTD method is widely used for high-frequency electromagnetic field analyses for designing electrical devices. The method is one of the iterative stencil computations composed of nested loops, both outer temporal and inner spatial. In iterative stencil computations, the number of floating point operations is often relatively small compared with the total amount of data transferred between CPUs and main memory. Consequently, the computational time of the 3-d FDTD kernel is often affected by the (effective) memory bandwidth rather than the performance of the processing core. In particular, on a computational node based on multi-core processors, the speed-up attained is seldom proportional to the number of cores used, despite the fact that the computational kernel of the 3-d FDTD method can be straightforwardly parallelized. This is because of the well-known "memory-wall problem". In iterative stencil computations, "time-space" tiling is one remedy for this problem [1].

In time-space tiling, the analyzed domain is divided into a multiple of small parts (tiles), each sufficiently small compared with the cache size; the electromagnetic field variables in each tile are updated for multiple time steps. Because most of the calculations on the tile are performed on the cache, the technique increases the cache hit ratio and thus exhibits better performance. The application of the technique to two-dimensional FDTD kernel was reported in [2]. However, because the 3-d FDTD kernel has a more complex stencil shape and loop structure than the two-dimensional case, reports on time-space tiling for the 3-d FDTD method are scant. Given this circumstance, we previously proposed the time-space tiling method for the 3-d FDTD kernel without the redundant calculation (3-d tiled FDTD), and confirmed its effectiveness on a multi-core processor system [3][4].

In this paper, to achieve further performance improvements, we describe a parameter tuning of the 3-d tiled FDTD. The tuned parameters are the shape of tiles and the number of time steps updated for a tile. We tune these parameters based on experimental results using relatively small-sized jobs in which the FDTD kernel is performed for a small number of time steps. It is shown that the tuned 3-d FDTD kernel attains 2.28-fold better performance than the naïve implementation of the kernel.

II. Parameter Tuning for 3-d Tiled FDTD Kernel

A. Time-space Tiling for 3-d FDTD Method

In the 3-d tiled FDTD method, we first divide the analyzed domain into multiple small regions (tiles). Although the tiles are sequentially processed in lexicographical order, the electromagnetic field variables are updated for multiple time steps in each tile. It is noted that the tile is moved by one grid point in all negative x-, y-, and z- directions in every time step to avoid redundant calculations. Figure 1 shows the movement of the tile and the order of calculations for each grid point. Our previous report [4] proved that the above process properly updates the field variables. For multithread parallel processing, the spatial loop for the field variable on each tile is parallelized by using OpenMP.

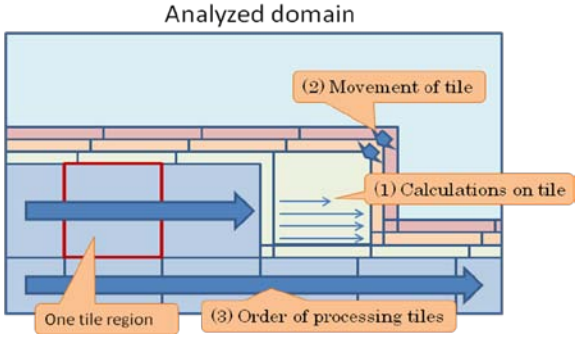

Fig. 1. Time space tiling for FDTD method

B. Method of Automatic Tuning

In the 3-d tiled FDTD kernel, the tuned parameters are given by the number of grid points of each tile in the x-, y-, and z-directions denoted by t_x, t_y, and t_z, respectively, and the number of time steps updated on each tile, t_s. Before determining the tuning strategy, we consider the general characteristics of the parameters on the performance of the kernel. Because the performance improvement by the tiling technique is derived from the increase in the cache hit ratio,

the tile size must be smaller than the cache size. Moreover, as the effect of increasing t_s gradually declines, the setting of too many time steps in a tile need not be examined. Therefore, we fix the maximum number for t_s at 30.

Next, we conducted preliminary numerical experiments to provide a general understanding of the effects of tuning the parameters. The numerical result implies that local optimization techniques, such as the steepest descent method, do not work well due to the existence of many local minima for the objective function given by the computational time. Moreover, t_s was found to be adjustable separately from the tile size. Considering these results, we propose the following tuning strategy:

STEP 1. (Cubic tile optimization) Fix t_s at 10. Optimize the tile size, t_c, subject to $t_c=t_x=t_y=t_z$. Let t_c' denote the optimized value obtained for t_c.

STEP 2. Optimize $t_s \in \{1,2,\ldots,30\}$ using the tile size $(t_c')^3$ with t_c' obtained in **STEP 1**. Here, let t_s' denote the optimized value of t_s.

STEP 3. (Tile shape optimization) Fix t_s at t_s'. Perform the Monte Carlo method where we only examine tiles fulfilling the following size criteria:

(a) $0.95\times(t_c')^3 \le t_x \times t_y \times t_z \le 1.35\times(t_c')^3$

(b) $t_y < t_x$, $t_y < t_z$

STEP 4. Optimize t_s using the tile size obtained in **STEP 3** and determine the final tuned parameters.

III. Numerical results

To examine the tuned 3-d FDTD kernel, we performed a numerical test on a multi-core processor. An 8-core Intel Sandy Bridge processor was used. The analyzed domain is set to 800^3 and the total number of time steps set at 90. In the tiled FDTD program, the computation in each tile is multi-threaded using OpenMP. The most outer loop, which is given by the spatial loop in the x-direction, is parallelized. Moreover, it is noted that the elements of the arrays for the electromagnetic field variables are contiguous in the z-direction.

To manage a number of test jobs for the tuning process, we used Xcrypt, which is a script language for job management [5]. Xcrypt is based on Perl and provides many useful functions for running many jobs on a parallel computer operated with a batch queuing system.

Table I provides the performance comparison of the tuned 3-d FDTD program and the program based on the naïve implementation using four nested loops for time and three directions in space. The tuned version clearly exhibits superior performance in both sequential and parallel computations. From the approximately 600 test jobs performed for the tuning process, the tuned kernel was about 2.28-fold faster than the naïve implementation on the parallel computation using 8 threads. Moreover, Figure 2 shows the relative (parallel) speedup compared with the sequential FDTD kernel based on the naïve implementation. Figure 2 confirms that the tuned kernel attains good parallel performance.

TABLE I. PERFORMANCE EVALUATION OF 3-D TILED FDTD KERNEL WITH TUNED PARAMETERS

(a) Sequential computation using one thread

	# test jobs	t_x, t_y, t_z, t_s	Elapsed time (s)
Naive implementation	-	-	1586
Tuned kernel by Step 2 (cubic tile)	103	48, 48, 48, 23	875
Tuned kernel	633	35, 8, 414, 29	686

(b) Parallel computation using eight threads

	# test jobs	t_x, t_y, t_z, t_s	Elapsed time (s)
Naive implementation	-	-	237
Tuned kernel by Step 2 (cubic tile)	96	40, 40, 40, 17	135
Tuned kernel	626	24, 10, 320, 15	104

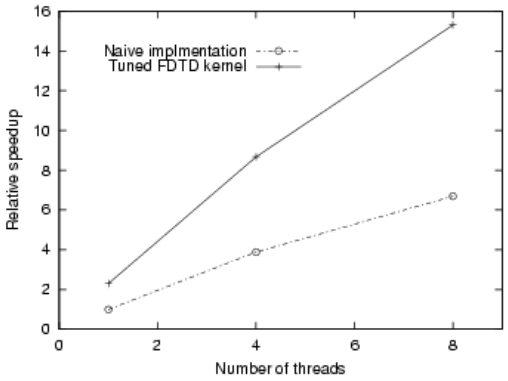

Fig. 2. Speedup compared with sequential computation using the naive implementation

IV. Conclusion

In this paper, we introduced the three-dimensional tiled FDTD method and proposed an auto-tuning technique of the tiling parameters to enhance performance. To tune the tile shape, we use the Monte Carlo method while taking the cache size into consideration. The resulting tuned FDTD kernel attains twice better performance than the kernel of the naïve implementation in both serial and parallel computations.

References

[1] M. Wolf, "More iteration space tiling", Proc. Supercomputing '89, pp. 655–664, (1989).

[2] R. Strzodka et al., "Cache oblivious parallelograms in iterative stencil computations", Proc. ICS '10, pp. 49–59, (2010).

[3] T. Minami et al., "Cache-Aware Performance Improvement of Three-Dimensional FDTD Kernel", IPSJ Tran. Advanced Computing Systems, vo.4, pp. 70-83, (2011), (In Japanese).

[4] T. Minami et al., "Temporal and Spatial Tiling Method without Redundant Calculations for Three-Dimensional FDTD Method", IPSJ Tran. Advanced Computing Systems, to appear, (In Japanese).

[5] T. Hiraishi et al., "Xcrypt: A Perl Extension for Job Level Parallel Programming", Proc. WHIST2012, (2012).