High Performance Parallel FDTD Computation by Using Vector Processor and CUDA

Wang Xuanyu

University of Electronic Science and Technology of China

June 4 2016

FDTD

FDTD: Finite Difference Time Domain.

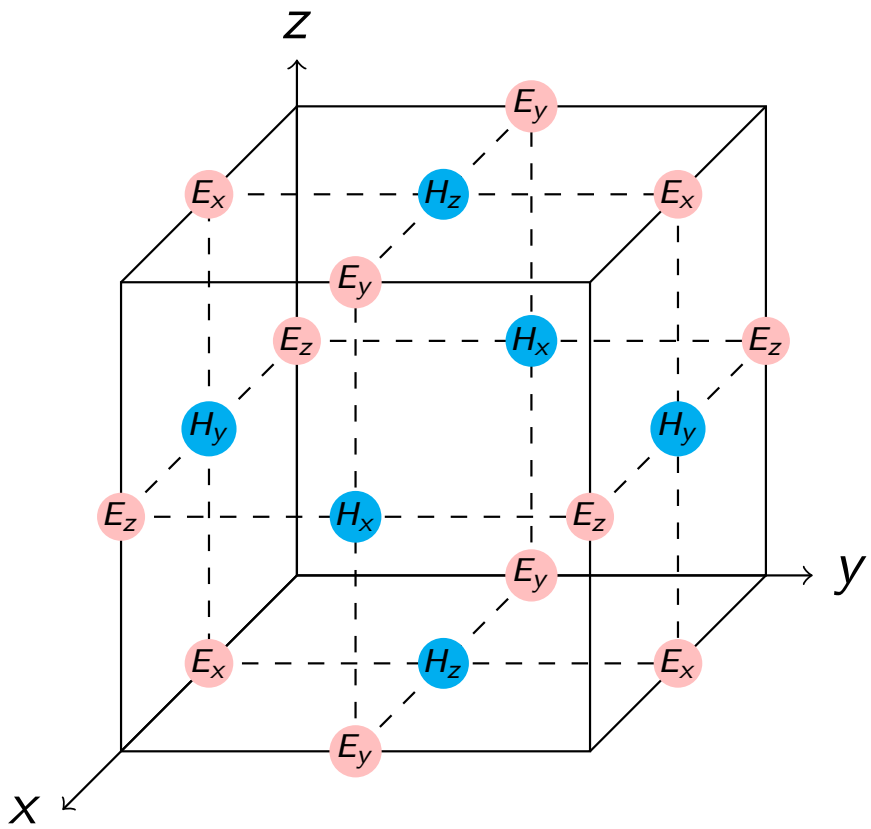

Figure: The spatial discrete structure of Yee cell

Vector Processor

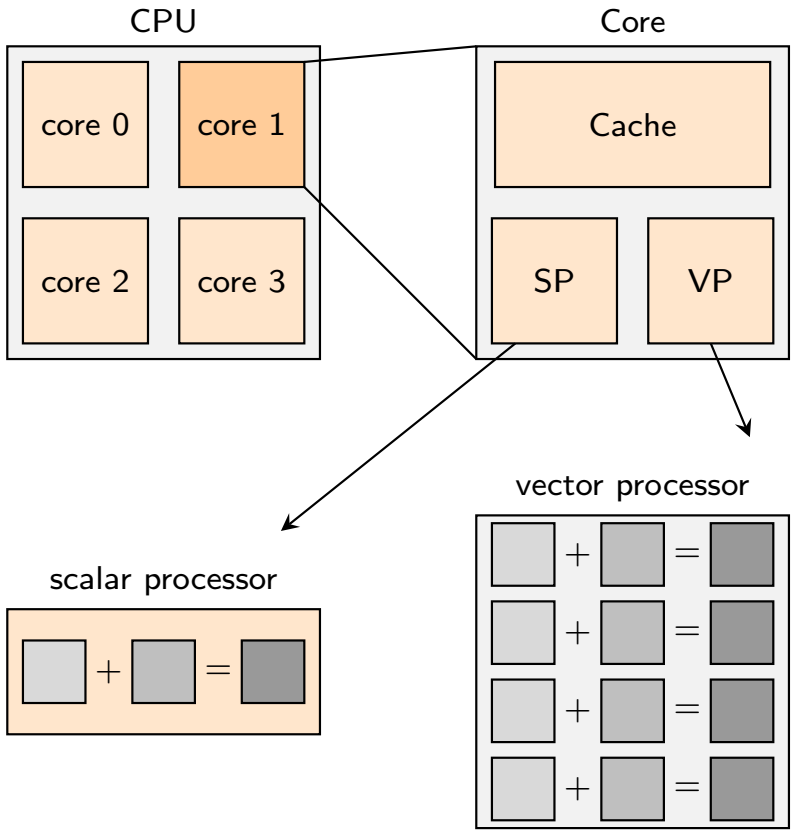

Figure: The spatial discrete structure of Yee cell

CUDA

CUDA: Compute Unified Device Architecture.
Characteristics:

- Massive threads.
- Independent device.

FDTD with VP

New model

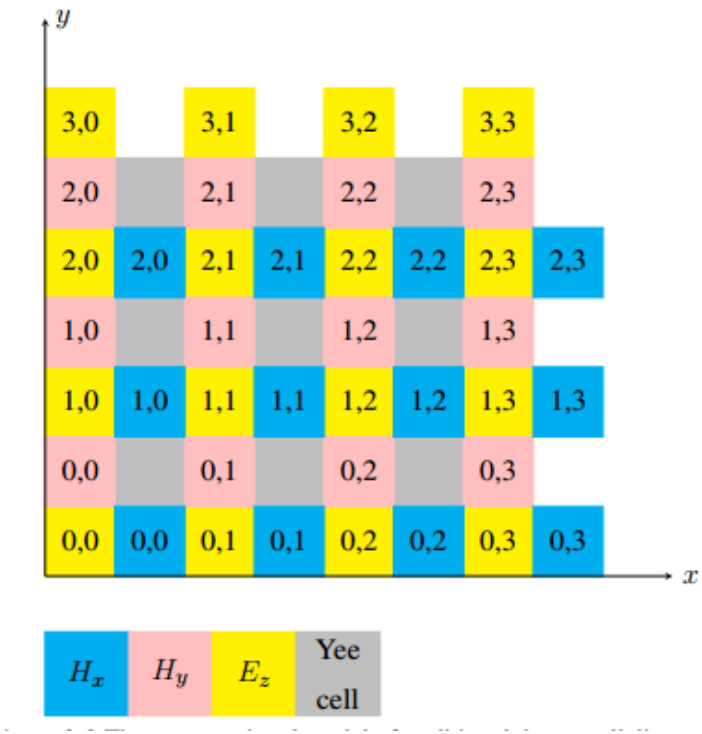

Figure: The traditional computational model

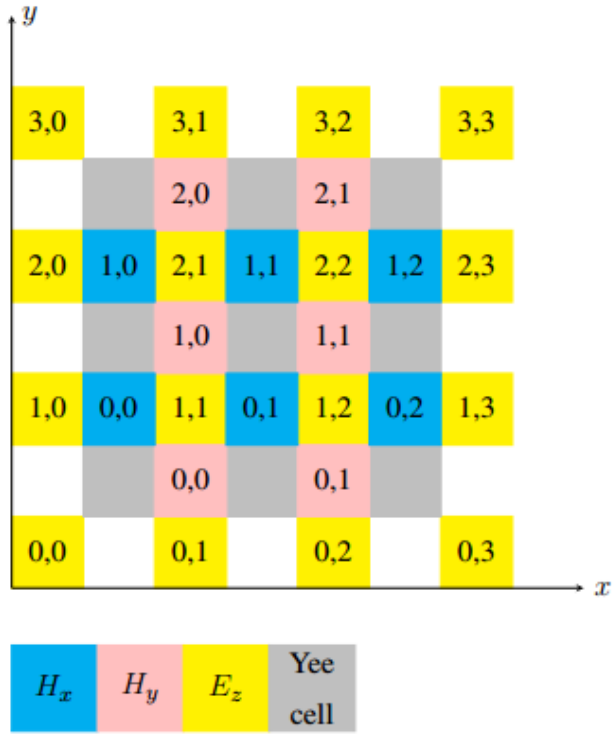

Figure: The modified computational model

Figure: The traditional and new computational model

FDTD with VP

Comparasions

Number of discrete field points

Field component	The number in x direction	The number in y direction
E_z	N_x+1	N_y+1
H_x	N_x+1	N_y
H_y	N_x+1	N_y

Table: The number of traditional scheme

Field component	The number in x direction	The number in y direction
E_z	N_x+1	N_y+1
H_x	N_x	N_y-1
H_y	N_x-1	N_y

Table: The number of modified scheme

FDTD with VP

Comparasions

Time elapsed

Function	Elapsed Inclusive (ms)		Time saved by using new model
	Traditional model	New model	
`main`	4229.46	4082.64	3.47%
`compute`	4216.81	4070.73	3.46%
`H_cmp`	2.50	2.41	3.37%
`E_cmp`	1.68	1.62	3.57%

[1] The size of simulation area is 1000×1000 Yee cells.

[2] The number of time step is 1000.

Table: The comparison between traditional and new computational model

FDTD with VP

In different conditions

The relation between elapsed time and simulation size:

Space size The size of space scale is as n times as before, the time-consuming will be about $0.92n + 0.08$ times than before.

Time size The size of time scale is as n times as before, the time-consuming will be about $0.99n$ times than before.

FDTD with CUDA

Implementation

```
1     int x, y, tid, number;
2     float dif_Hy, dif_Hx;
3     tid = threadIdx.x + blockIdx.x*blockDim.x;
4     while (tid < ele_ex*size_Ez_y)
5     {
6         number = tid + 1;
7         y = number % ele_ex;//row
8         x = number - (y*ele_ex);//column
9         //Hy(i,j)    -    Hy(i-1,j)
10        dif_Hy = Hy[y*ele_hy + x] - Hy[(y - 1)* ele_hy + x];
11        //Hx(i,j-1) -    Hx(i,j)
12        dif_Hx = Hx[y*ele_hx + (x - 1)] - Hx[y*ele_hx + x];
13        Ez[y*ele_ex + x] += coe_Ez * (dif_Hx + dif_Hy);
14        tid += blockDim.x*gridDim.x;
15    }
```

FDTD with CUDA

Comparison

Function	Elapsed Inclusive (ms)		The time saved by using CUDA
	The modified data parallelism	Using CUDA	
`main`	7835.78	888.30	88.67%
`H_cmp`[3]	4.64		
`E_cmp`[3]	3.13		
`Hy_cmp_kernel`[4]		<0.01	
`Hx_cmp_kernel`[4]		<0.01	
`Ez_cmp_kernel`[4]		<0.01	

[1] The size of simulation area is 2000×1000 Yee cells.

[2] The number of time step is 1000.

[3] Only in serial way.

[4] Onely in the way of using CUDA.

Table: The comparison between the modified data parallelism and using CUDA

FDTD with CUDA

In different conditions

In all conditions, time elapsed in a single running time is less that 0.01.

Conclusion

In this, we did following contributions:

FDTD with VP Proposed a new computational model, which can save about 3.45% time. The result had been sent to a journal.

FDTD with CUDA Implemented the Mur ABC with CUDA. In the profiling result we can see how powerful the GPU is in parallel computation.

Acknowledgements

Thanks for your patience.