Proceedings of 20th International Congress on Acoustics, ICA 2010

23-27 August 2010, Sydney, Australia

On Decreasing the Calculation Time in Multi-Dimensional Acoustic Numerical Simulation by Multi-GPU Parallel Computing

Naoki Kawada (1), Kan Okubo (1), Norio Tagawa (1) and Takao Tsuchiya (2)

(1) Faculty of System Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino-shi, Tokyo 191-0065, Japan
(2) Faculty of Science and Engineering, Doshisha University, 1-3 Tatara-Miyakodani, Kyotanabe-shi, Kyoto 610-0321, Japan

PACS: 43.58.Ta, 43.55.Ka

ABSTRACT

Numerical analysis for sound wave propagation in time domain has been investigated widely as a result of computer development. Now, the development of accurate numerical schemes in time domain is an important technical issue. When we analyze large-scale sound wave propagation, the reduction of the calculation time is a necessary requirement. Recently, GPU (Graphic Processing Unit) is used as an acceleration tool for the calculation in various study fields. This movement is called GPGPU (General Purpose computing on GPUs). In the last few years the performance of GPU keeps on improving rapidly. This study makes an examination on decreasing the calculation time in acoustic field numerical analysis using GPU. We implement time-domain acoustic simulation (FDTD method, CIP method, and GCIP method) on GPU by CUDA (Compute Unified Device Architecture).

INTRODUCION

To date, numerical analysis for sound wave propagation in time domain has been investigated widely as a result of computer development. Acoustic simulation in time domain is an effective technique for the estimation of time-series sound pressure data (e.g., nonlinear acoustic propagation phenomenon, acoustical measurements and instrumentation, architectural acoustics). Now, the development of accurate numerical schemes in time domain is an important technical issue.

The finite difference time domain (FDTD) method [1-8] is the most popular scheme used in time-domain acoustic simulation. However, we know that, using Yee's leapfrog algorithm, finite difference approximation certainly causes error owing to numerical dispersion. In the past study, the authors have proposed an acoustic simulation technique using generalized constrained interpolation profile method (GCIP method) [9], which is an expanded CIP method [10-16]. It is a method of characteristic [17] with very high accuracy; i.e., it enables the calculation with less-numerical dispersion. However, this method requires more calculation time than the conventional dispersive schemes.

When we analyze large-scale sound wave propagation, the reduction of the calculation time is a necessary requirement. Of course, there are usually trade-off relationships between required calculation time and numerical accuracy. Generally, calculation time and computational cost are proportional to the number of grid points. Additionally, more accurate schemes also require more computational cost.

Recently, GPU (Graphic Processing Unit) is used as an acceleration tool for the calculation in various study fields [18]. This movement is called GPGPU (General Purpose computing on GPUs) [19]. In the last few years the performance of GPU keeps on improving rapidly. That is, a PC (personal computer) with GPUs might be a personal supercomputer. GPU computing gives us the high-performance computing environment at a lower cost than before. Therefore, the use of GPUs contributes to a significant reduction of the calculation time in large-scale sound wave propagation.

This study makes an examination on decreasing the calculation time in acoustic field numerical analysis using GPU. We implement time-domain acoustic simulation (FDTD method, CIP method, and GCIP method) on GPU by CUDA (Compute Unified Device Architecture) [20-21]. We examine suitable algorithm and efficient thread models of CUDA for single- and multi- GPU computing.

ACOUSTIC NUMERICAL SIMULATION

Governing equations for linear acoustic fields

The governing equations for linear acoustic fields are given in Eq. (1) and Eq. (2).

$$\rho \frac{\partial \vec{v}}{\partial t} = -\nabla p \, , \tag{1}$$

$$\nabla \cdot \vec{v} = -\frac{1}{K}\frac{\partial p}{\partial t} \, . \tag{2}$$

In those equations, ρ denotes the density of the medium, K is the bulk modulus, p is sound pressure and v is the particle velocity. Here we assume that the calculation is for a lossless and homogeneous medium.

Moreover, for simplicity, assuming $\vec{v} = (v_x,\ 0,\ 0)$ in order to analyze one-dimensional (1-D) acoustic field propagation in the x-direction, we can obtain the following equations from Eq. (1) and Eq. (2).

$$\frac{\partial v_x}{\partial t} + \frac{1}{\rho}\frac{\partial p}{\partial x} = 0, \tag{3}$$

$$\frac{\partial p}{\partial t} + K\frac{\partial v_x}{\partial x} = 0. \tag{4}$$

FDTD method

We can obtain Eqs. (5) and (6) from Eqs. (3) and (4) by employing second-order central difference approximation on a staggered grid.

$$v_x^{n+\frac{1}{2}}(i+\frac{1}{2},j) = v_x^{n-\frac{1}{2}}(i+\frac{1}{2},j) - \frac{\Delta t}{\rho}\frac{p^n(i+1,j)-p^n(i,j)}{\Delta x}, \tag{5}$$

$$p^{n+1}(i,j) = p^n(i,j) - K\Delta t\frac{v_x^{n+\frac{1}{2}}(i+\frac{1}{2},j) - v_x^{n+\frac{1}{2}}(i-\frac{1}{2},j)}{\Delta x}, \tag{6}$$

where Δt is the timestep, and Δx is the grid size. Figure 1 depicts the grid model used in three-dimensional (3-D) FDTD analysis.

GCIP(*l,m*) method

By addition and subtraction of Eq. (3) and Eq. (4), we obtain

$$\frac{\partial(p \pm Zv_x)}{\partial t} \pm c\frac{\partial(p \pm Zv_x)}{\partial x} = 0. \tag{7}$$

In those equations, Z indicates the characteristic impedance (i.e. $Z = \sqrt{\rho K}$) and c represents the sound velocity in medium (i.e. $c = \sqrt{K/\rho}$).

In addition, through simple spatial differentiation of the equations, the derivatives are given as

$$\frac{\partial(\partial_x p \pm Z\partial_x v_x)}{\partial t} \pm c\frac{\partial(\partial_x p \pm Z\partial_x v_x)}{\partial x} = 0. \tag{8}$$

Therein, $\partial_x = \partial/\partial_x$ Figure 2 depicts the grid model used in GCIP analysis, in which both acoustic field components and the derivatives of fields are located on the same grid (i.e., collocated grid).

We show the means to calculate the fields of the (n + 1) time step from the fields of n time step, applying the CIP method to discretized acoustic field components. We define F_{x+}, F_{x-}, G_{x+} and G_{x-} as follows:

$$F_{x+} = p + Zv_x, \tag{9}$$

$$F_{x-} = p - Zv_x, \tag{10}$$

$$G_{x+} = \partial_x p + Z\partial_x v_x \tag{11}$$

$$G_{x-} = \partial_x p - Z\partial_x v_x. \tag{12}$$

Consequently, these components on grid points ($x = i\Delta x$) at the time step n are given F_{x+}, F_{x-}, G_{x+} and G_{x-} as

$$F_{x\pm}^n(i) = p^n(i) \pm Zv_x^n(i), \tag{13}$$

$$G_{x\pm}^n(i) = \partial_x p^n(i) \pm Z\partial_x v_x^n(i). \tag{14}$$

Next, applying the Hermite interpolation (H and H') to $F_{x\pm}^n(i)$ and $G_{x\pm}^n(i)$ yields the following equations related to propagation to the $\pm x$ -direction.

$$F_{x\pm}^{n+1}(i) \leftarrow \mathrm{H}^{(l,m)}(F_{x\pm}^n, G_{x\pm}^n), \tag{15}$$

$$G_{x\pm}^{n+1}(i) \leftarrow \mathrm{H}^{(l,m)}(F_{x\pm}^n, G_{x\pm}^n). \tag{16}$$

Here, $\mathbf{H}^{(l,m)}$ represents the Hermite interpolation with the l-th order using additional m moments. For example, $\mathbf{H}^{(3,1)}$ expresses the Hermite interpolation with the 1st order using physical values and their derivatives. That is, Hermite inter

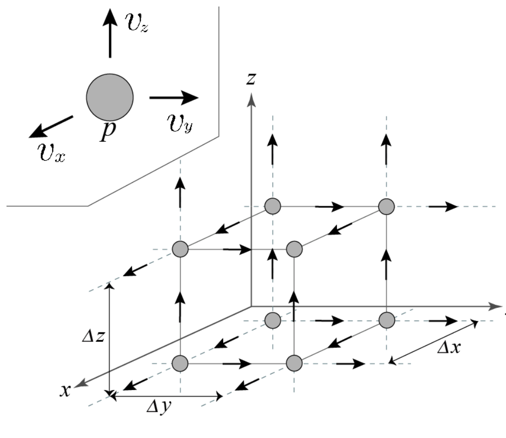


Figure 1. FDTD grid model

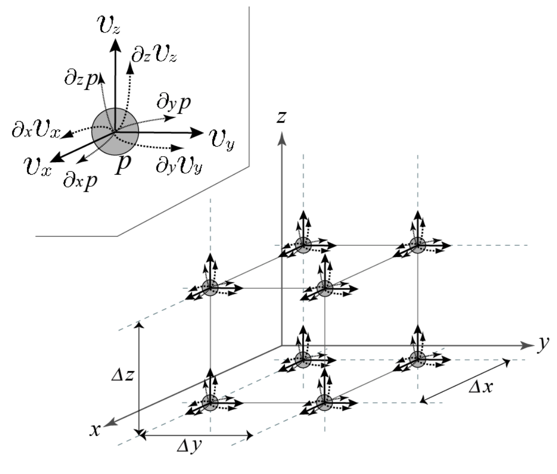


Figure 2. GCIP grid model

polation in case of the GCIP(3,1) method, represented as $\mathbf{H}^{(3,1)}$, is given as

$$F_{x\pm}^{n+1}(i) = \sum_{k=i\mp1}^{i} h_k^{(0)}(x) F_{x\pm}^{n}(k) + \sum_{k=i\mp1}^{i} h_k^{(1)}(x) G_{x\pm}^{n}(k), \tag{17}$$

$$G_{x\pm}^{n+1}(i) = \sum_{k=i\mp1}^{i} h'^{(0)}_k(x) F_{x\pm}^{n}(k) + \sum_{k=i\mp1}^{i} h'^{(1)}_k(x) G_{x\pm}^{n}(k). \tag{18}$$

On the other hand, Hermite interpolation $\mathbf{H}^{(7,1)}$ used in the GCIP(7,1) method is given as

$$F_{x\pm}^{n+1}(i) = \sum_{k=i\mp2}^{i\pm1} h_k^{(0)}(x) F_{x\pm}^{n}(k) + \sum_{k=i\mp2}^{i\pm1} h_k^{(1)}(x) G_{x\pm}^{n}(k), \tag{19}$$

$$G_{x\pm}^{n+1}(i) = \sum_{k=i\mp2}^{i\pm1} h'^{(0)}_k(x) F_{x\pm}^{n}(k) + \sum_{k=i\mp2}^{i\pm1} h'^{(1)}_k(x) G_{x\pm}^{n}(k), \tag{20}$$

where $h_k^{(m)}(x)$ is assumed Hermite interpolation function. That is, the present method involves the use of the values of the acoustic field and their spatial derivatives at two or four grid points.

Moreover, using the following Eqs. (21) to (24), one can obtain acoustic field components (p and v_x) of time step (n+1).

$$p^{n+1}(i) \leftarrow \frac{F_{x+}^{n+1}(i) + F_{x-}^{n+1}(i)}{2} \tag{21}$$

$$v_x^{n+1}(i) \leftarrow \frac{F_{x+}^{n+1}(i) - F_{x-}^{n+1}(i)}{2Z} \tag{22}$$

$$\partial_x p^{n+1}(i) \leftarrow \frac{G_{x+}^{n+1}(i) + G_{x-}^{n+1}(i)}{2} \tag{23}$$

$$\partial_x v_x^{n+1}(i) \leftarrow \frac{G_{x+}^{n+1}(i) - G_{x-}^{n+1}(i)}{2Z} \tag{24}$$

GPU PROGRAMMING BY CUDA

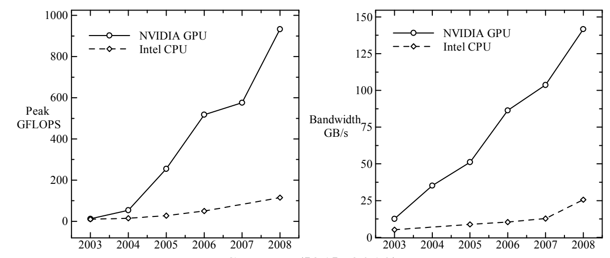
Recently, GPU has evolved into a highly parallel, multi-threaded, many core processor with great computational ability and very high memory bandwidth owing to the huge computing demand for real-time and high-definition 3D graphics [4]. Figure 3 shows the performance comparison between GPUs and typical CPUs.

In November 2006, NVIDIA introduced CUDA, a general purpose parallel computing architecture. CUDA comes with a software environment that allows developers to use C as a high-level programming language. GPU should be used as an acceleration tool for the calculation in computational acoustics.

Figure 4 depicts CUDA's grid model and block model. The blocks of the kernel grid, which is invoked by a CUDA program on the host CPU, are enumerated and distributed to multiprocessors with available execution capacity. The threads of a thread block execute concurrently on one Streaming Multiprocessor (SMP). As thread blocks terminate, new blocks are launched on the vacated SMP.

A SMP consists of eight Streaming Processor (SP) cores, a multithreaded instruction unit, on-chip constant cache, texture cache and shared memory. GPU have several SMPs; for example, GeForce GTX 285 owns 30 SMPs, GeForce GTX 295 owns 2 × 30 SMPs. That is, GeForce GTX 285 has 240 parallel cores. Figure 5 shows the Hardware Model of NVIDIA GPU (Compute Capability 1.0 – 1.3).

We can use multiple GPUs programming as CUDA devices, if the computer system loads same type GPUs. Multiple GPUs computation needs to transfer boundary data between neighboring nodes across the PCI Express bus. Therefore, in multiple GPUs programming, we must consider the overhead introduced by the transference.



Source: ([21], 2010)

(a) Peak FLOPS (b)Memory Bandwidth

Figure 3. Performance comparisons between GPUs and typical CPUs [4] (Source: ([21], 2010))

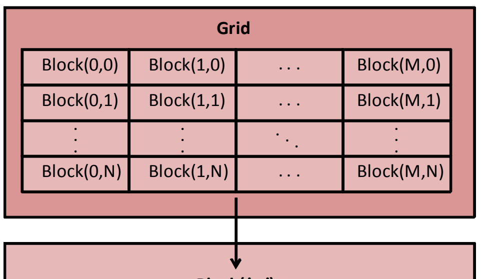


Figure 4. Thread Hierarchy (Grid of Thread Blocks) (Source: ([21], 2010))

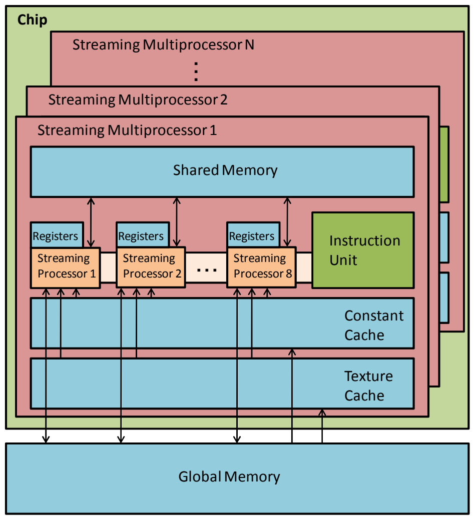


Figure 5. Hardware Model of NVIDIA GPU (Compute Capability 1.0 – 1.3) (Source: ([21], 2010))

RESULTS

NVIDIA's CUDA programming environment is used with NVIDIA Geforce GTX 285 and GTX 295 GPUs. For comparison, we employ the PC with Intel Core i7 920 2.67GHz. This processor has eight hyperthreaded cores, or effectively scales 8 threads. We use FLOPS (Floating point number Operations per Second) and FUPS (Fields Update Per Second) for performance evaluation of GPU calculation. 1 FUPS means the capability of updating acoustic fields point once per second. In this study, we use single precision floating-point numbers.

We adopt a simple uniform media (i.e., air) as a calculation model. Uniform square grids are employed for CPU and GPU calculation. We calculate the pressure distribution at each time. Moreover, in FDTD analysis we don't consider absorbing boundary condition, whereas GCIP analysis automatically sets this condition.

2-D Acoustic Field Calculation (using single GPU)

We estimated the calculation time required for a 2-D simple acoustic model except for the absorbing boundaries. First, we show calculation time using single Geforce GTX 285 GPU. Table 1, 2, and 3 show results of comparison of the calculation time between the GPU and CPU results, where the analysis region is 1024 × 1024 cells and whole calculation time is divided into 1024 time steps. Here, all measurements are made on the above 285 GPU and Intel Core i7 processor machine with 12 GB of memory.

Table 1. Calculation Time of FDTD method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	3.23	3.99	0.33
GTX 285	0.34	37.90	3.16

Table 2. Calculation Time of GCIP(3,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	57.58	2.69	0.019
GTX 285	2.06	75.06	0.52

Table 3. Calculation Time of GCIP(7,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	100.66	2.90	0.011
GTX 285	2.12	137.76	0.51

These tables also provide results of FLOPS and FUPS. FDTD results using GPU is ca. 9.5 times faster than 8-threads CPU calculation. The GCIP(3,1) method using GPU is ca. 28 times faster than 8-threads CPU calculation. Moreover, the GCIP(7,1) method using GPU is ca. 47.5 times faster than CPU results. Relative comparison of the run-time between the CIP result and FDTD result reveal the following: CIP analysis requires more run-time than FDTD analysis when equivalent discretization is used. However, by using GPU calculation, difference of required calculation time between FDTD result and CIP result became small.

2-D Acoustic Field Calculation (using multi-GPUs)

Next, we show results of acoustic field calculation using multiple GPUs. Table 4 shows the comparison results of FDTD calculation between the GPU results and CPU result, where the analysis region is 8192 × 8192 cells and whole calculation time is divided into 1024 time steps. All measurements are made on the above 295 GPU and Intel Core i7 processor machine. This table gives the calculation time in case of 1-GPU, 2-GPUs, 4-GPUs, and 8-GPUs. This result illustrated FDTD calculation on 8 GPUs is ca. 48 times faster than 8-thread CPU calculation.

Table 5 and 6 show the comparison results of GCIP calculation between the GPU results and CPU result, where the analysis region is 3072 × 3072 cells and whole calculation time is divided into 1024 time steps. All measurements are made on the above 295 GPU and Intel Core i7 processor machine. These tables also give the calculation time in case of 1-GPU, 2-GPUs, 4-GPUs, and 8-GPUs. This result reveals the followings: GCIP(3,1) calculation on 8 GPUs is ca. 117.5 times faster than 8-thread CPU calculation. On the other hand, GCIP(7,1) calculation on 8 GPUs is ca. 114.7 times faster.

Next, Fig. 6, 7, and 8 respectively show the calculation time by FDTD method and GCIP methods against the number of grids in the case of 1-GPU, 2-GPUs, 4-GPUs, and 8-GPUs computing. This indicates that the overhead introduced by the data transference between multi-GPUs caused the increase of calculation time when the number of grids is small. On the

Table 4. Calculation Time of FDTD method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	176.91	4.66	0.39
*GTX 295 * 1*	26.07	31.63	2.64
*GTX 295 * 2*	13.33	61.86	5.16
*GTX 295 * 4*	6.81	121.09	10.09
*GTX 295 * 8*	3.68	224.09	18.67

Table 5. Calculation Time of GCIP(3,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	415.95	3.35	0.023
*GTX 295 * 1*	20.13	69.13	0.48
*GTX 295 * 2*	10.58	131.53	0.91
*GTX 295 * 4*	5.63	247.17	1.72
*GTX 295 * 8*	3.54	393.10	2.73

Table 6. Calculation Time of GCIP(7,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	540.01	4.87	0.018
*GTX 295 * 1*	22.19	118.46	0.44
*GTX 295 * 2*	11.98	219.41	0.81
*GTX 295 * 4*	6.66	394.67	1.45
*GTX 295 * 8*	4.71	558.07	2.05

other hand, the calculation time almost linearly increases with the increase of the number of grids if it becomes larger. That is because operation time is dominant as compared to data transference time with the increase of simulation area.

We show results of calculation time in case of the large-scale analysis using multi-GPUs. Table 7 shows FDTD results, where the analysis region is 23552 × 23552 cells and whole calculation time is divided into 1024 time steps. FDTD calculation on 8 GPUs is 52.8 times faster than multi-thread calculation on CPU.

Table 8 and 9 show GCIP results, where the analysis region is 10240 × 10240 cells. GCIP(3,1) calculation on 8 GPUs is 153 times faster than 8-thread calculation on CPUs. On the other hand, GCIP(7,1) calculation on 8 GPUs is 179.6 times faster than 8-thread calculation on CPU. As a comparison of table 5 and 6 with table 8 and 9, we can see the multi-GPUs calculation by GCIP methods is effective for the analysis of the large-scale region.

Table 7. Calculation Time of FDTD method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	1463.16	4.66	0.39
*GTX 295 * 8*	27.72	245.89	20.49

Table 8. Calculation Time of GCIP(3,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	4532.48	3.41	0.024
*GTX 295 * 8*	29.62	522.01	3.63

Table 9. Calculation Time of GCIP(7,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	6185.25	4.72	0.017
*GTX 295 * 8*	34.44	848.02	3.12

3-D Acoustic Field Calculation (using single GPU)

Table 10 shows results of comparison of the calculation time for 3-D acoustic simulation between the GPU and CPU results, where the analysis region is 256 × 256 × 256 cells and whole calculation time is divided into 1024 time steps. Here, all measurements are made on the above 285 GPU and Intel Core i7 processor machine. It is clarified that 3-D FDTD acoustic calculation using GPU is ca.16.5 times faster than 8-threads CPU calculation. Therefore, in single GPU FDTD calculation, 3-D acoustic analysis is more accelerated than 2-D analysis.

Table 11 and 12 show the comparison results of GCIP calculation between the GPU results and CPU result, where the analysis region is 128 × 128 × 128 cells and whole calculation time is divided into 1024 time steps. These tables indicate the followings; the 3-D GCIP(3,1) acoustic calculation using GPU is ca. 37 times faster than 8-threads CPU calculation. Moreover, the GCIP(7,1) method using GPU is ca. 53 times faster than CPU results. In single GPU GCIP calculation, 3-D acoustic analysis is more accelerated than 2-D analysis, similarly.

Table 10. Calculation Time of FDTD method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	115.20	2.68	0.15
GTX 285	6.98	44.30	2.46

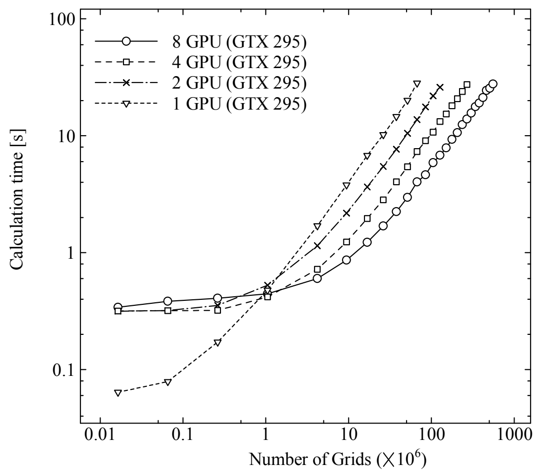


Figure 6. Calculation Time (FDTD method)

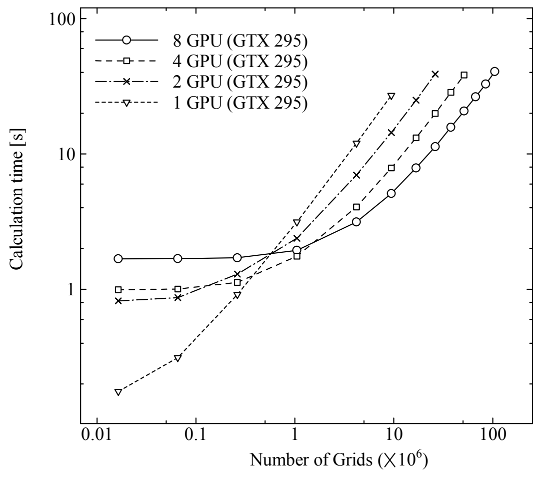


Figure 7. Calculation Time(GCIP(3,1) method)

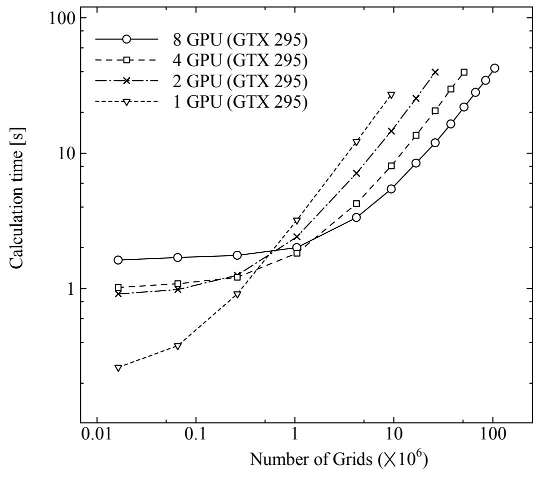


Figure 8. Calculation Time (GCIP(7,1) method)

3-D Acoustic Field Calculation (using multi-GPUs)

Next, we show results of 3-D acoustic field calculation using multiple GPUs. Figure 9 depicts the present division model for parallel computing. We divide the analysis domain intomultiple sub-domains along *z*-direction. The number of sub-domains is corresponding to that of multi-GPUs. Table 13 shows the comparison results of FDTD calculation between the GPU result and CPU result, where the analysis region is 512 × 512 × 512 cells. All measurements are made on the above 295 GPU and Intel Core i7 processor machine.

Table 11. Calculation Time of GCIP(3,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	466.17	1.99	0.0046
GTX 285	12.62	73.51	0.17

Table 12. Calculation Time of GCIP(7,1) method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	702.03	2.50	0.0031
GTX 285	13.27	132.05	0.16

This table gives the calculation time in case of 8-GPUs. This result illustrated FDTD calculation on 8 GPUs is ca. 59 times faster than 8-thread CPU calculation.

Figure 10 shows the calculation time by FDTD method against the number of grids in the case of 8-GPUs computing. In this figure, the results of following four cases are shown: (1) case X; analysis domain expands along *x*-direction, (2) case Y; analysis domain expands along *y*-direction, (3) case Z; analysis domain expands along x, *z*-direction, (4) case XYZ; analysis domain equally expands along *x*, *y*, and *z*-direction. These results illustrates that analysis domain should expand along the division direction.

CONCLUSIONS

This study made an examination on decreasing the calculation time in numerical analysis for sound wave propagation in time domain using single GPU and multi-GPUs computing. As a result, it is clarified that multi-GPU FDTD calculation is faster than multi-CPU FDTD calculation by 48 times at the maximum in 2-D acoustic analysis. On the other hand, multi-GPU GCIP calculation needs less calculation time than multi-CPU calculation by about 1/100 times. Therefore, the multi-GPUs calculation is very effective for GCIP methods as an acceleration tool.

Moreover, for large-scale acoustic simulation, these results show the feasibility of high-speed and high-precise simulation analysis by hardware acceleration using multi-GPU calculation. In near future we intend to examine more effective thread model for 3-D GPU calculation.

Table 13. Calculation Time of FDTD method

Device	Calc Time[s]	GFLOPS	GFUPS
Core i7(8 threads)	836.97	2.96	0.16
*GTX 295 * 8*	14.17	174.59	9.70

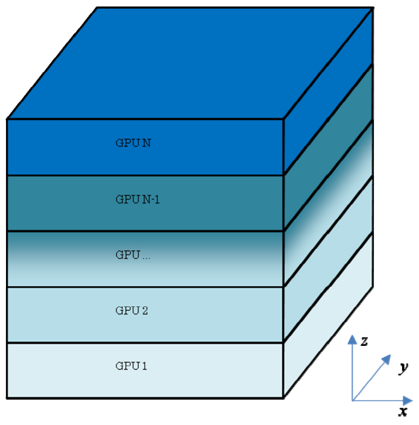


Figure 9. Division model

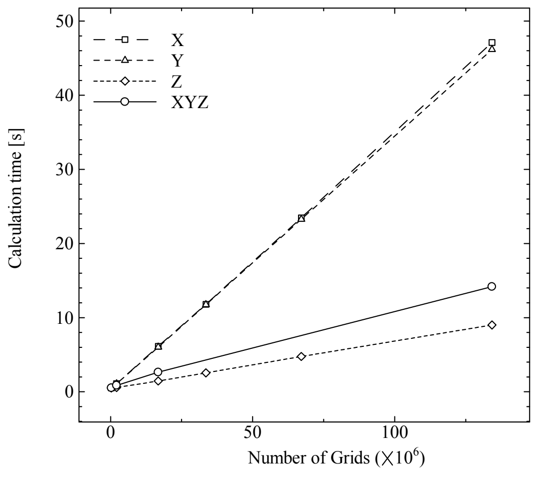


Figure 10. Calculation Time versus number of grids

REFERENCES

1 K. S. Yee: IEEE Trans. Antennas Propag., vol. AP- 14, no. 4, pp. 302-307, May 1966.
2 K. S. Kunz and R. J. Luebbers, The Finite Difference Time Domain Method for Electromagnetics, CRC Press, 1993.
3 J. Virieux: Geophysics, vol. 49, no. 11, pp. 1933–1942, Nov., 1984.
4 Zhang, C., LeVeque, R.J.: Wave Motion, vol. 25, no.3, pp. 237–263, 1997
5 Kudo, H., Kashiwa, T., Ohtani, T.: vol.85, no.9, pp. 15–24, 2002.
6 Sendo, Y., Kudo, H., Kashiwa, T., Ohtani, T.: Electronics and Communications in Japan, vol.86, no.11, pp. 30–37, 2003.
7 Sakamoto, S., Seimiya, T., Tachibana, H.: Acoustical Science and Technology vol.23, no.1, pp. 34–39, 2002
8 M. Sato: Jpn. J. Appl. Phys. 46 (2007) 4514 .
9 K. Okubo, T. Tsuchiya, R. Seta, N.Tagawa, IEEE Ultrasonics Symposium, Rome, 2009
10 H. Takewaki, A. Nishiguchi and T. Yabe: J. Comput. Phys., vol. 61, pp. 261–268, 1985.
11 T. Yabe, X. Feng and T. Utsumi: J. of Comput. Phys., vol. 169, pp. 556–593, 2001.
12 K. Okubo and N. Takeuchi: IEEE Trans. Antennas Propag., vol. 55, No. 1, pp.111-119, Jan. 2007.
13 K. Okubo, S. Oh, T. Tsuchiya and N. Takeuchi: IEICE Trans. Fundamentals, vol. E90-A, No. 9, pp.2000-2005, Sept. 2007.
14 M. Konno, K. Okubo, T. Tsuchiya, and N. Tagawa,: Jpn. J. Appl. Phys., 48 (2009) 07GN01
15 T. Tsuchiya, K. Okubo and N. Takeuchi: Jpn. J. Appl. Phys., 47 (2008) pp. 3952-3958
16 M. Konno, K. Okubo, T. Tsuchiya and N. Tagawa: Jpn. J. Appl. Phys. , 47 (2008), pp.3962-3963
17 G. D. Smith, Numerical Solution of Partial Differential Equations, Oxford University Press, 1965.
18 T. Tsuchiya and H. Sekoguchi, Journal of the Japan Society for Simulation Technology, 27, 4, pp.245-254, 2008.
19 http://gpgpu.org/
20 www.nvidia.com/object/cuda_home.html
21 http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf