Acceleration of Numerical Analysis for Acoustics by FDTD for Structural Health Monitoring System

Kiyoshi Hayakawa^{a,*}, Akira Oki^a, Toshitaka Umemoto^a, Takeshi Wada^a, and Takashi Obata^b

^aOsaka Prefecture University College of Technology, 26-11 Saiwai-cho, Neyagawa-shi 572-8572, Japan

^bHokkai-Gakuen University, 4-1-40 Asahi-cho, Sapporo-shi 062-8605, Japan

*Corresponding Author: hayakawa@osaka-pct.ac.jp

Abstract

We are developing a structural health monitoring system for infrastructures (i.e. bridge). We are going to employ wave numerical analysis for acoustics by FDTD (Finite Difference Time Domain). However, FDTD needs huge compute resource and time for large area analysis such as bridges. In order to reduce the compute time, we constructed a 20 nodes computer cluster called SAFHC (Smart ARM+FPGA Hetero Cluster). In this paper, we reported FDTD parallelization on SAFHC. SAFHC consists of ZYBO and Open Blocks built in ARM processor, is able to execute program with low power consumption. In an evaluation, we achieved 14.8 times faster at 20 nodes.

Keywords: Acoustics Wave analysis, Graphic Simulation, MPI, Open-MP.

1. Introduction

Most infrastructures, such as office buildings, bridges, high ways, and so on, were built in 1970's and 1990's in Japan. Deterioration of the infrastructures in 1970's becomes a social problem. Conventionally, the infra-structures were demolished and rebuilt. However, affected by the recent Japanese economic slump, the infrastructures were repaired and fix. Furthermore, we require the durability and earthquake-proof of the infrastructures for largescale disasters and increase in car traffic. Consequently, the infrastructures become lifelong duration.

In order to put it into practice, it is important to check and manage the building structures on a daily basis. Ordinary, the building structures are checked by hammering test of specialists. In order to check the numerous old infrastructures, we need an automatically hammering test

system. In the hammering test system, we analyze hammering acoustics wave on the infrastructures by FDTD (Finite Difference Time Domain). It is easy to execute FDTD program in parallel. Generally, FDTD is applied into simulation of Electro dynamics modeling techniques. It transforms Maxwell's equation into difference equations which are discretized to time domain and space domain. However, it also transforms acoustics wave equation into difference equations.

We are developing a low power computer cluster called SAFHC. For the computer clusters set in small companies, it is necessary to fully consider power consumption and the space occupied by the cluster. PC clusters require larger and more fans to cool down the CPUs. Therefore, PC clusters that use low-power processors have been developed⁽¹⁻⁶⁾. SAFHC also employ dual core ARM which is one of the low power processor and FPGA (ZYBO board). In the future, we execute FDTD program using ARM and FPGA.

In this paper, we make reference to ARM execution only, and evaluate data arrangement and communication methods of FDTD. We measure execution time and communication time of FDTD and speedup ratio including the number of the compute nodes.

2. Numerical Analysis for Acoustics by FDTD

2.1 Analysis by FDTD

FDTD is one of Electrodynamics modeling techniques. Since it is a time domain method, FDTD solutions can cover a wide frequency range with a single simulation run, and treat nonlinear material properties in a natural way.

In order to analyze acoustics wave, we use differential equations of an acoustics wave (motion and continuity). The equations represent relation between particle velocity and

acoustic pressure (1) (2) and differential equation of acoustic pressure (3). Where, U indicates particle velocity, P indicates acoustic pressure, ρ indicates volume density, and k indicates bulk modulus.

$$\frac{\partial U}{\partial t} = -\frac{1}{\rho} \times \frac{\partial P}{\partial x} \quad (1)$$

$$\frac{\partial P}{\partial t} = -k \frac{\partial U}{\partial x} \quad (2)$$

$$P_z^n = \frac{2 + \Delta t}{2 - \Delta t} \times p_z^{n-1} - \frac{2k}{2 - \Delta t} \times \frac{\Delta t}{\Delta x} \quad (3)$$

2.2 Absorbing Boundary Condition

In FDTD simulation, waves protruding to an analysis region are reflected and interfere with non-reflected waves. In order to prevent form the interference, we set an absorbing boundary condition. It absorbs the wave which protruding to the analysis region on the end of region. We employ Mur absorbing boundary condition⁽⁷⁾.

2.3 Incident Wave

We employ pulse wave using gauss function as an incident wave. Gauss function represents formula (4) below. Where, μ indicates x value of center of pulse, σ indicates width of the incident wave. In order to make pulse wave, σ is set as small as possible.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (4)$$

3. SAFHC

3.1 System Organization

Fig. 1 depicts the organization of SAFHC system that is constructed as a test bed of an ARM and FPGA cluster working environment. It consists of ARM+FPGA nodes (4 nodes), ZYBO node (16 nodes), an electric power management system, a data base server for the electric power management and a host computer for the calculations.

The ARM+FPGA nodes and ZYBO nodes are used for low electric power calculation notes. They execute applications for large data and high performance such as FDTD CIP, and Monte Carlo simulation.

There are two networks on SAFHC, primary network and secondary network. The calculation nodes, the host computer, the database server, and power management system are connected with primary network. The ARM processor connects the FPGA calculation device by secondary network. Secondary network for the ARM+FPGA nodes employ Ethernet and for the ZYBO nodes employ AXI of AMBA bus. We put tree topology into these hierarchy networks for Molecular Orbital Calculation.

Fig. 1. Organization of SAFHC

4. Acceleration

4.1 Shared Memory and Distributed memory

Dual core ARM embedded into ZYBO and Open blocks is connected by shared memory with AXI bus. Therefore, organization inside of the chip is shared memory multi-processor. As the programing environment on shared memory multi-processor, we employ Open MP. We insert directives of Open MP into a program source.

On the other hands, each ZYBO and Open blocks is directly connected by Ethernet. There isn't any shared memory among ZYBO and Open blocks. Therefore, organization of among ZYBO and Open blocks is distributed memory multi-processor. As the programing environment on distributed memory multi-processor, we employ MPI.

4.2 Parallelization by Open MP

Open MP is implemented in Intel C/C++ compiler of gcc 4.1 later. We insert directives (Fig.2) into FDTD program.

In the execution of FDTD program, ARM core execute program with single thread before a directive point, and execute it with multi thread after the directive point.

```
#pragma omp parallel for
for( i=0 ; i<N ; i++ ){
    /* calculation */
}
```

Fig. 2. An Example of Directive

4.3 Parallelization by MPI

MPI is message passing interface library. Generally, data used by an application are divided into the number of nodes, and assign to each nodes. In Acoustics FDTD, a node which has adjacent data sends them to next to the node, and vice versa. Therefore, each node share data of the border (adjacent data). For example on the Fig.3, node-0 sends adjacent data to node-1, and node-1 sends another adjacent data. Acoustics FDTD performs the exchanging shared adjacent data after the calculation of the acoustic pressure is finished.

Fig. 3. Sharing Adjacent Data

4.3.1 Communication Method for Shared Adjacent Data

In order to send/receive the shared data, we use peer to peer communication by MPI. There are two peer to peer communication on MPI such as blocking or non-blocking. We employ blocking communication due to implement it easily. Blocking communication uses MPI_send MPI_receive only. MPI guarantees blocking communication to complete send and receive process. That is, MPI_send waits until MPI_receive is completed and MPI_receive waits until a node complete receiving the data. Therefore, MPI doesn't care about completion of the communication. On the other hand, blocking communication occurs a lot of waiting points.

4.3.2 Data Distribution to The Nodes

It is important for FDTD calculation to divide the analysis region properly. We evaluate two region segmentation methods, 1 dimension method (See Fig.4 (a) : Call 1-D) and 2 dimension method (See Fig.4 (b) : Call 2-D).

Since 2 dimension method is more communication than 1-D, communication latency of 2-D longer than that of 1-D. But communication time of 2-D is less than that of 1-D.

Fig. 4. Region Segmentation Methods

We measure communication time of these methods, and compare them. The analysis region is $2.304E-7[m^2]$. We divide it into 1200×1200 . Fig. 5 shows relation between communication time and the number of the nodes.

Fig. 4. Region Segmentation Methods

In case of 4 nodes, it assumes that there are a lot of communication collisions. Communication time of 2-D is longer than that of 1-D on 4 nodes. In case of over 8 nodes, communication collisions decrease due to reduce the number of communications by one node. Therefore, we employ 2-D region segmentation method.

4.4 Acoustics by FDTD Program

Fig.5 shows program flow chart of the acoustics FDTD program. To begin with, Node 0 makes the program start as host node on MPI.

Node 0 initializes the program such as assignment of the data, then makes another nodes execute the program. Dual core ARM in each node calculate acoustic pressure in parallel with Open MP. They also calculate absorbing boundary condition after complete acoustic pressure calculations. They share the adjacent data with MPI blocking communication. They calculate particle velocity using data of acoustic pressure which they have including shared adjacent data. In the calculation of particle velocity, they use dual core ARM with Open MP.

After calculation of the particle velocity is over, they increment time clock by one. They calculate acoustic pressure again using particle velocity before one time clock. If analysis time (Tmax) is over, each node send data of acoustic pressure and particle velocity to host node (Node 0), and finish the program.

Fig. 5. Flow Chart of acoustic FDTD program

5. Evaluation

5.1 Simulation Result

Fig.6 shows graphic simulation result of acoustic wave propagation by FDTD analysis. We got the same result of dual cores and PC. The analysis area of the acoustic FDTD program is $2.304\text{E-}7[\text{m}^2]$. The analysis time is $1.2[\text{s}]$. The analysis area divided into 400×400 X-Y cells. One time clock

is $6[\text{ms}]$. The vehicle of the acoustic wave is water. The volume density is $1000[\text{kg}/\text{m}^3]$.

Fig. 6. Acoustics Wave Propagation Simulation

5.2 Execution Time Results

Fig.7 shows Execution time of the acoustic wave propagation simulation by FDTD analysis. We measured execution time of the program described Section 4.4 by using MPI_Wtime() function. We measured elapsed time of each function such as pressure, boundary, share, velocity, and gather function. We divided these functions into communication functions or calculation functions. Calculation time includes elapsed time of pressure, boundary, and velocity functions. Communication time includes share and gather functions.

In terms of total execution time, the time of 20 nodes cluster is 14.8 times as fast as 1 node. In terms of calculation time, 20 nodes cluster achieved super linear speed up (36.3 times speed up). On the other hands, communication time on 20 nodes cluster increased 1.57 times as long as 4 nodes. Communication time was longer than calculation time over 16 nodes. A ratio of communication time in the entire execution time was about 59% on 20 nodes.

Fig.7 Relation between the number of node and execution

In order to consider the parallel efficiency on our cluster system in detail, we show elapsed time of each functions described table 1 and Fig.8.

It took a constant time to calculate absorbing boundary condition and gather calculation results regardless of the number of nodes. The elapsed time of calculation of acoustic pressure and particle velocity reduced with an increase the number of nodes. Specially, time reduction rate of particle velocity is larger than that of calculation of acoustic pressure. It assumes that the reason of the large reduction rate is that memory cache hit rate of calculation data increase due to increasing entire cache size. It took almost same time in 4 nodes and 8 nodes on share function for exchange of adjacent data. The elapsed time of it was suddenly rise on 16 nodes. The elapsed time of it on 16 nodes was almost same as 20 nodes.

Table 1. Elapsed time of each functions

		The number of nodes			
	Function	4	8	16	20
Elapsed Time[s]	Pressure	73.8	46.0	22.9	18.3
	Boundary	1.7	1.5	1.3	1.2
	Share	42.2	42.1	62.1	65.5
	Velocity	95.1	64.3	32.5	26.2
	Gather	0.47	1.54	1.41	1.69

Fig. 8. Elapsed time of each functions

6. Related works

In terms of low-power components in cluster systems, Anderson et al. have proposed fast array of wimpy nodes (FAWN) system as a cluster architecture for low-power data intensive computing⁽⁹⁾. FAWN combines low-power CPUs with small amounts of local flash storage, and balances computation and I/O capabilities in order to provide efficient

parallel data access on a large-scale. SAFHC also consists of wimpy nodes, but FAWN employ Intel chip, and we are going to make an integration system of ARM and FPGA.

ASTRO is CPU + FPGA hybrid platforms which is based on detection of Micro Blaze instruction sequences by execution profiling on a simulator, followed by synthesis of one loop accelerator per candidate instruction sequence. ASTRO focuses on maximizing memory access parallelism. Dynamic memory access analysis is performed to determine disjoint regions of access. SAFHC also aim to be a CPU + FPGA hybrid platforms. But SAFHC is different from ASTRO in terms of power management.

7. Conclusions

In this paper, we investigated parallelization of acoustic wave analysis by FDTD using Open MP and MPI. We constructed SAFHC which consists of 20 nodes hetero cluster. Each node has dual core ARM and FPGA. We use dual core ARM only. Dual core ARM in each node calculate acoustic pressure in parallel with Open MP. Sharing method for adjacent data employs MPI non-blocking communication.

In the evaluations, we measured execution time of FDTD program. In terms of total execution time, the execution time of 20 nodes cluster was 14.8 times as fast as 1 node. However, the evaluation shows that our cluster system may not reduce execution time over 20 nodes due to increase of communication time.

There are two future works in term of energy efficient cluster computing.

One is improvement of communication method in order to reduce execution time of FDTD over 20 nodes. We have to improve communication protocol of Ethernet. We employ TCP/IP communication protocol. Context switch occurs with considerable frequency (about 10×10^3 times), when communication operation on Linux copies communication data from user memory to communication buffer on Linux kernel in TCP/IP protocol. In order to reduce the number of context switch, we will employ RDMA (Remote Direct Memory Access).

The other is a hybrid hetero calculation using ARM and FPGA. We have implemented a multi-core processor on an FPGA for FDTD which is one of explicit time stepping method⁽¹⁰⁾. We simulated electromagnetic field by FDTD and evaluated execution time, circuit size and integral power consumption. We will integrate this paper's work and the previous work into one.

Acknowledgment

This work is supported in part by Grant No. 26420472 (study of central control of health monitoring for structures using cloud computing) from Japan Society for the Promotion of Science.

References

- (1) Warren, W., Weigle, A. et al.: "High Density Computing A 240 Node Beowulf in One Cube Meter. Super Computing." CD-ROM, 2002.
- (2) D. Strenski, :The Cray XD1 computer and its reconfigurable architecture, Cray, Tech. Rep., 2005.
- (3) R. Baxter et al., :Maxwell – a 64 FPGA supercomputer, Engineering Letters, vol. 16, no. 3, pp. 426–433, 2008.
- (4) M. Showerman et al., :QP: A heterogeneous multiaccelerator cluster, in 10th LCI International Conference on High-Performance Clustered Computing, Boulder, Colorado, 2009.
- (5) Takayuki Imada, Mitsuhsa Sato, Yoshihiko Hotta and Hideaki Kimura.: Power Management of Distributed Web Servers by Controlling Server Power State and Traffic Prediction for QoS, HPPAC, 2008.
- (6) Kiyoshi Hayakawa, Keita Ito, : EXECUTION ENVIRONMENT ON FPGA FOR SMART PC HETERO-CLUSTER, 23rd IASTED International conference Parallel and Distributed Computing and Systems, pp.469–475, 2011.
- (7) Mur, G. : "Absorbing boundary conditions for the finite difference approximation of the time-domain electromagnetic-field equations", IEEE Trans. Electromagn Compat, Vol. EMC-23, No. 4, 377–382, 1981.
- (8) Naoyuki Shida, Shinji Sumimoto, Atsuya Uno, : "MPI Library and Low-Level Communication on the K computer", FUJITSU Sci. Tech. J., Vol. 48, No. 3, pp. 324–330, 2012
- (9) D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. "FAWN: A Fast Array of Wimpy Nodes. Commun". ACM, 54(7), 101–109, 2011.
- (10) Lin, Mingjie and Chen, Shaoyi and DeMara, Ronald F. and Wawrzynek, John, "ASTRO", Microprocess. Microsyst., Vol.39, Num.7, pp.553–564, 2015.
- (11) Kiyoshi Hayakawa, Ryusuke Yamano, "Multi-Core FPGA Execution for Electromagnetic Simulation by FDTD", 2nd International Conference on Information Science and Control Engineering, pp.829–833, 2015