Distributed Simulator for General Control System in CEMTool

Tairi Lee*, Young Sam Lee, Kwan Ho Lee, and Wook Hyun Kwon

*School of Electrical Engineering & Computer Science, Seoul National University, Seoul, Korea
(Tel : +82-2-880-7314; E-mail: trlee@cisl.snu.ac.kr)

Abstract: This paper proposes a distributed simulator for general control system in CEMTool. Systems can be described by SIMTool likes the simulink in Matlab. For distributed simulation, we can seperate any system into several parallel subsystems in SIMTool. The number of parallel subsystem can be determined by the system's property. After seperation, parallel simulator will do initialization, one-step-ahead simulation, block-distribution and ordering and so on. Finally, simulator will create independent C codes and executive files for each subsystem. The whole system is fulfilled by several PCs, and each PC executes one subsystem. There are communications among these subsystem using reflective memory or ethernet. We have made several experiments, and the 5-stand cold rolling mill control system is our main target. The result of parallel simulation has shown effective speedup in comparison with one pc simulation.

Keywords: Distributed simulation, Simulator, Parallel, C Code, Speedup, CEMTool, SIMTool

1. INTRODUCTION

Distributed simulation or parallel simulation deals with wide range problems and the ultimate object of distributed simulation is to improve the computation speed of systems. Many researchers have dealt with the concepts and algorithms of parallel simulation.[1][2][4] Even though, there are a few results about system simulation example, but their objects are not for computation speedup.[3]

Therefore, in the current paper, a new approach have made to realize the distributed simulation for dynamical systems, and the main result of this approach is computation speedup. For dynamical systems, data communication and synchronization must be taken in each step of distributed simulation.

To attain the computation speedup of distributed simulation for dynamical systems, we will build system with a block diagram editor called SIMTool which likes the simulink in matlab, and divide the system into several subsystems, and then take the simulation and computation in CEMTool. SIMTool and CEMTool are popular tools in many universities of Korea.

The speed of data communication is very important for the distributed simulation of dynamical system since the system need data communication in each step. If the time of communication is much larger than that of computation, it is hard to realize speedup. To shorten the time of communication, we use reflective memory as the media of communication.

Distributed simulator is a universal simulator for any dynamical system. we construct a system model called 5 stand cold rolling mill as our main target since 5 stand cold rolling mill system has more than 2 thousands blocks and this system need long computation time and enough complexity to demonstrate the common use of distributed simulator.

To execute high speed simulation of dynamical system, we propose a structure of multiple C code generation. After dividing the whole system into several subsystems in SIMTool, we generate C code for each subsystem and compile these C codes to executive files finally. One subsystem simulation is executed in one PC and there are data communication among these subsystems with reflective memory in each step. This paper will show that distributed simulation can bring the speedup by dividing the the 5 stand mill system.

The overall introduction for distributed simulation in CEMTool is described in Section 2. In Section 3, the detailed structure and the procedure of the whole multiple C code generation is discussed. In Section 4, experiments show the results of distributed simulation. Finally, the conclusion is given in Section 5.

2. Introduction for distributed simulation

Distributed simulation mainly includes three parts. They are system division, multiple C code generation, execution.

First, the system division is done in SIMTool. The division process divide the system into several parallel blocks, and one parallel block represents one subsystem. Ordinary, there are some connections between two subsystems, including real connection and virtual connection. Real connection is made by connection line and virtual connection is made by from and goto blocks.

Second, multiple C code generation is made in CEMTool after system division in SIMTool. SIMTool notifies CEMTool the direction of distributed simulation by DDE communication, and the information of divided system are included in out file. For example, the system named pblk.blk is divided to three subsystems as described in Fig 1.

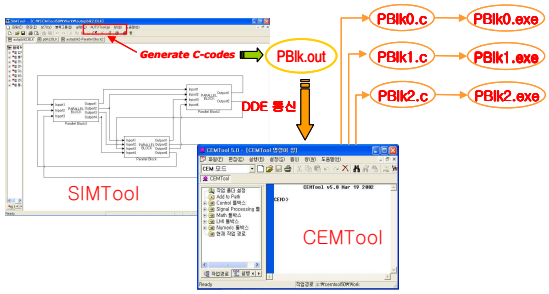


Fig. 1. The basic procedure for the parallel simulation in SIMTool and CEMTool

SIMTool creates system information file pblk.out and CEMTool reads this out file when receives the order by DDE communication. Since there are three subsystems, CEMTool generates pblk0.c, pblk1.c and pblk2.c, these C files can be compiled to executive files pblk0.exe, pblk1.exe and pblk2.exe.

The multiple C code generation is made by distributed simulator in CEMTool, which includes following processes, initialization, one-step-ahead simulation, block distribution, ordering and final parallel block C code generation.

Finally, these three executive files fulfilled in three PC and there are data communications in each simulation step as seen in Fig2. The media for data communication can be reflective memory or ethernet, but there is speedup only when media is reflective memory.

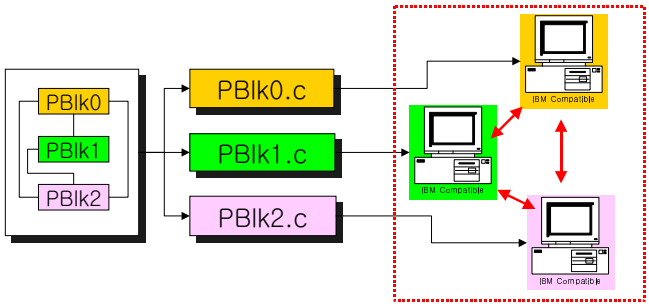


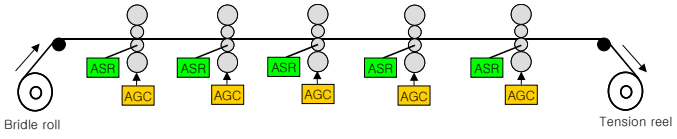
Fig. 2. 5 stand cold rolling mill system

3. Realization of distributed simulation

Starting discussion of distributed simulation, 5 stand mill system is introduced as a target model. And then describe the detailed process of distributed simulation using this system model.

3.1. 5 stand cold rolling mill system

5 stand cold rolling mill is a metal working process whereby strip or sheet metal is deformed by successive rolling operation. Each rolling operation is achieved by squeezing the strip between a set of rolls. To achieve the required reduction and final tolerance for any given product, several rolling operations will be required and these are done in tandem to achieve high production volumes of specific product and result in the high precision of thickness while a single stand rolling mill operation is suitable for achieving low production volumes of a variety of products. The 5 stand cold rolling mill system is shown in Fig 3.



AGC: Automatic Gauge Controller
ASR : Automatic Speed Regulator

Fig. 3. 5 stand cold rolling mill system

We have described the 5 stand mill system in SIMTool as Fig 4. It consists of plant, controller and setup, but the computation loads of three parts are not well-balanced, plant holds more than 60% loads. In this paper, we divide the system into these three subsystems, so our speedup of distributed simulation is not very high.

5 stand mill system is composed of more than 2 thousands blocks, start time of 0, finish time of 40, time step of 0.001.

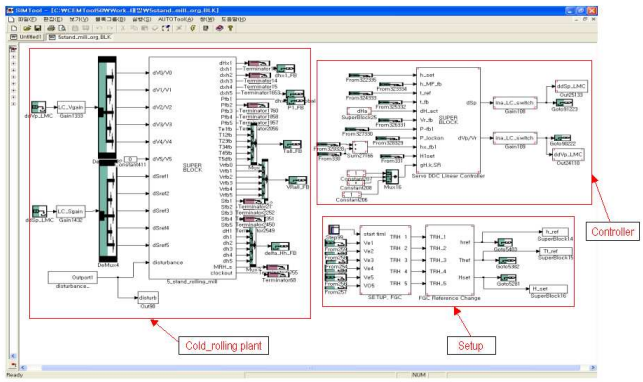


Fig. 4. 5 stand mill system described in SIMTool

3.2. System division in SIMTool

SIMTool is a block diagram editor for modelling dynamical systems and interacts with CEMTool for simulating and analyzing them. A block diagram modelling a dynamical system is drawn by first dragging adequate blocks from block libraries to the canvas, secondly connecting them with lines and finally setting the block parameters of each block.

SIMTool is just a block diagram editor and simulation is performed in CEMTool. Communication mechanism between SIMTool and CEMTool is a Microsoft Windows DDE protocol where DDE stands for dynamic data exchange.

If we separate the whole system into several subsystems using parallel blocks, communication ports named ComInport and ComOutport will be generated for data communication among systems. Fig5 is the result of 5 stand mill system distribution.

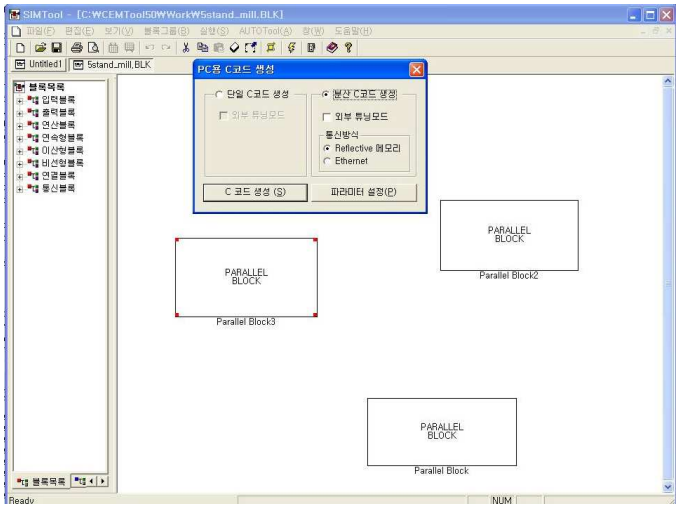


Fig. 5. 5 stand mill system described in SIMTool

To generate multiple C code, select autotool configuration

item in autotool menu of SIMTool. Then we can see the menu window as Fig5. There are radio buttons for single C code or multiple C code generation. If we select single C code generation, CEMTool just generate one set of C code for one PC simulation. We can make choice for multiple C code generation to execute parallel simulation.

From Fig5, there is no connection line between any two subsystems of 5 stand mill system because the subsystems are linked by virtual connection with From and Goto blocks. From and Goto blocks find their connection counterpart using tag whose property is classified into global and local. Local means Goto block and From block are in the same Superblock while Global means they can be in any Superblock. When create out file, SIMTool get the flat canvas of target system, which means dissolute the Superblocks and connect the virtual connection From and Goto blocks. In parallel simulation, if From and Goto blocks are not in the same parallel block, they are converted to communication port ComOutport and ComInport.

Also we can select the communication media from reflective memory and ethernet as seen in Fig5.

3.3. Multiple C code generation in CEMTool

At the beginning of this section, we compare the result of single C code generation and that of multiple C code generation.

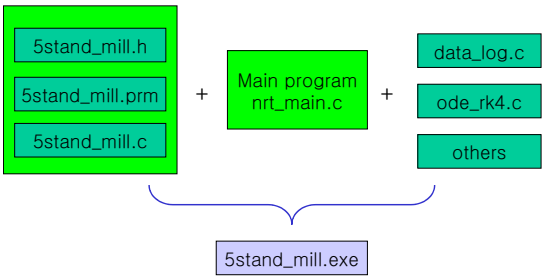


Fig. 6. one PC simulation for 5 stand mill system

At the case of single C code generation for 5 stand mill system, 5stand_mill.c, 5stand_mill.prm and 5stand_mill.h are generated and compile these files with nrt main and other required files into 5stand_mill.exe as described in Fig6, and then using this exe file we can do high speed simulation in one PC.

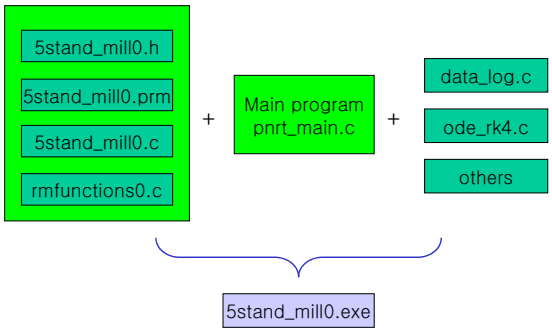


Fig. 7. distributed simulation for 5 stand mill system

When multiple C code generation for 5 stand mill system, suppose that the system is separated into three parts, 5stand_mill0.c, 5stand_mill0.prm and 5stand_mill0.h are generated for the first subsystem, and 5stand_mill1.c, 5stand_mill1.prm, 5stand_mill1.h, 5stand_mil2.c, 5stand_mil2.prm and 5stand_mil2.h are also formed. Besides these files, reflective memory communication files rmfunctions0.c, rmfunctions1.c and rmfunctions2.c are generated. As described in Fig7, first set of files can be compiled into 5stand_mill0.exe. The process of distributed simulator in CEMTool generation mainly includes initialization, one-step-ahead simulation, block distribution, ordering and multiple C code generation.

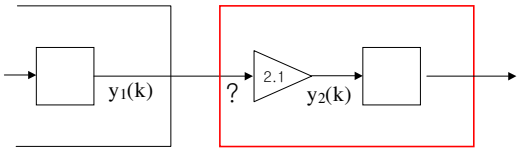


Fig. 8. reason of one-step-ahead

First, one-step-ahead simulation is necessary for distributed simulation. As described in Fig8, y_1 and y_2 belong to different parallel blocks. In kth step, to compute the value of $y_2(k)$, we need the value of $y_1(k)$. If there are no one-ahead-step simulation, there is only the value of $y_1(k-1)$. Simply speaking, at first step, to compute the value of $y_2(1)$, we need the value of $y_1(1)$. So we take the simulation for one step, and initialize the communication ports of subsystems.

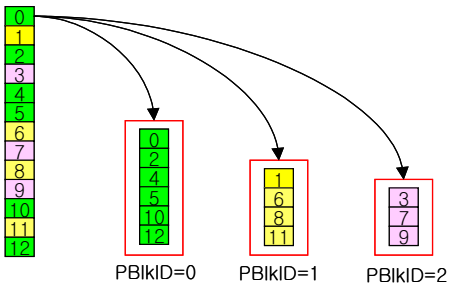


Fig. 9. block distribution

Second, CEMTool create arrays for subsystems, and all blocks are put into these arrays according to there parallel block number PBlkID. All these information is from out file created by SIMTool.

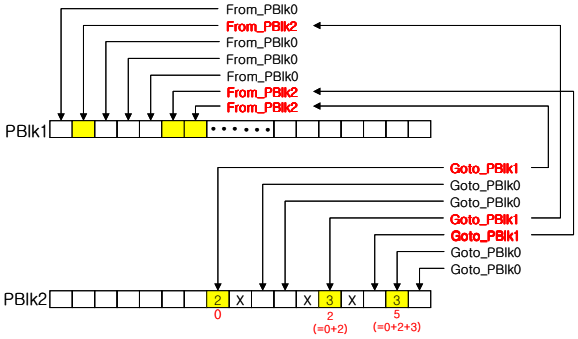


Fig. 10. block distribution

Third, ordering blocks is also very important for distributed

simulation. Wrong ordering blocks may result miscalculation and halt of data communication. ComInport has the highest priority in ordering and ComOutport has the lowest one. Fig10 reveals the offset computation for data communication by computing communication port data size.

Finally, we can start the C code generation after former steps. Parallel simulator includes .c file generator, header file generator, parameter generator, communication file generator, batch file generator.

After the executive files have been made for each subsystems, we start the distributed simulation using reflective memory. The flow chart of distributed simulation with reflective memory is Fig11.

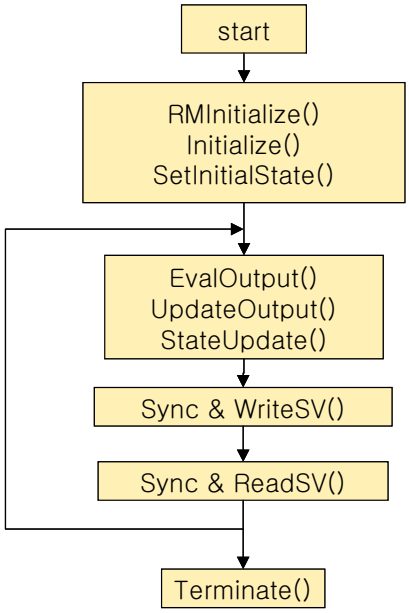


Fig. 11. flow chart of distributed simulation with reflective memory

In distributed simulation, reflective memory initialization, system and state initialization are made first, then the parallel simulation get into computation loop. Each step for subsystem consists of updating inner value of subsystem and communicating with other subsystems. Before writing and reading, there must be synchronization for all subsystems because subsystem write or read data at fixed address of reflective memory.

Distributed simulation with ethernet also can be realized. But the communication time with ethernet is too long to get speedup. So using ethernet is just a experience of distribution simulation for who don't have reflective memory.

4. Results

In our experiment, we selected 3 PCs with reflective memory. The 3 PCs are linked with optical cables and their CPU speeds are 750MHz, 500MHz and 400MHz respectively.

When we choose single C code generation and make the simulation in one PC, the simulation times are 30seconds, 52seconds and 63seconds respectively.

Table1 is the time list of distributed simulation for 5 stand mill system. Since the speeds of tested PCS are ill-balanced and the computation loads of subsystems of 5 stand mill are also out of balance, the speedup is not so high. But we can see that there is time saving at least, and the speedup in this distributed simulation is about 1.5 because the computation load of plant is over 60% of the 5 stand mill system.

Table 1. simulation time of distributed simulation for 5 stand mill system

Time	CPU1(750MHz)	CPU2(500MHz)	CPU3(400MHz)
20s	plant	controller	setup
22s	plant	setup	controller
38s	controller	plant	setup
40s	controller	setup	plant
45s	setup	controller	plant
38s	setup	plant	controller

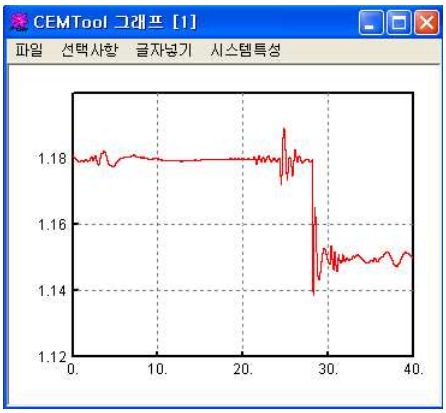


Fig. 12. the thickness value of 5 stand mill system

The accuracy of computation results of distributed simulation is also very important. Fig12 is the graph of 5th thickness of 5stand mill system. The results of distributed simulation agree with that of those of one PC simulation.

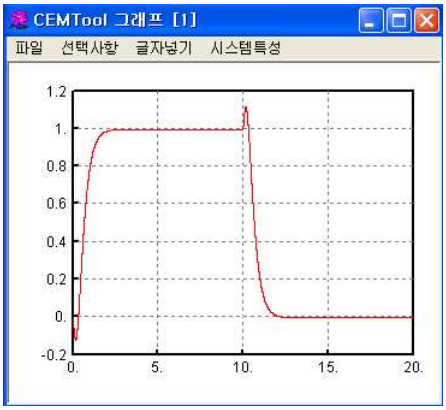


Fig. 13. distributed simulation for inverted pendulum

Another experiment is made with inverted pendulum system which is very sensible to outside disturbance. Fig13 is the graph of distributed simulation of inverted pendulum. The

results of distributed simulation also agree with those of one PC simulation.

5. Conclusion

In this paper, distributed simulation for dynamic system is presented. A system can be separated into several subsystems in SIMTool, and then generate multiple C code for those subsystems. Our main object of distributed simulation is to get speedup through contributing computation load to several PCs. The results show that there is definite time saving for 5 stand cold rolling mill system which have more than 2 thousands blocks. The time of data communication is very important for distributed simulation and directly affect the simulation time. In this paper, we use reflective memory as communication media that has high speed data transmission.

References

[1] Kai Hwang and Zhiwei Xu, *Scalable Parallel Computing,* McGraw-Hill, 2000.

[2] C.D.Pham, *Comparison of Message Aggregation Strategies for Parallel Simulations on a High Performance Cluster,* Modeling, Analysis and Simulation of Computer and Telecommunation Systems, 2000.

[3] L.Pollini and M.Innocenti, *A synthetic environment for dynamic systems control and distributed simulation,* IEEE Control Systems Magazine, vol.20, pp.49-61, April 2000.

[4] Barry Wilkinson and Michael Allen, *Parallel Progamming,* An Alan R.Apt Book, 1999.