Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

PERFORMANCE EVALUATION OF HIERARCHICAL DISTRIBUTED SIMULATORS[1]

Doo-Kwon Baik
Dept. of Computer Science
Wayne State University
Detroit, Michigan 48202

Bernard P. Zeigler
Dept. of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona 85721

ABSTRACT

A model can be decomposed into components, then mapped onto a network of simulators for execution. With different levels of decompositions, we are interested in finding the optimal model decomposition that can be implemented on a particular structure of distributed simulators. This is done by comparing performance measures such as the minimum response time or the maximum throughput per unit of hardware complexity. We claim that the need for a model capable of performance evaluation is an important concern in designing distributed simulators. Thus a methodology for performance evaluation and simulation modelling of hierarchically specified distributed simulators is required. From the empirical results, we found that simulating a given distributed simulator model is suitable to get optimal performances, that is, each model has an optimal decomposition level in terms of performance. The influential performance measures are the average of simulation run times and the product of the average run time and the hardware complexity of the given simulator model.

INTRODUCTION

Distributed simulation increases the speed of simulation by processing discrete event simulators in parallel. In the implementation of discrete event simulations, since a single processor takes maximum processing time, the number of processors can be traded-off against the time required for the distributed simulation [3]. Commonly, performance is not seriously considered until the system is built and many systems have unacceptable performance when completed. If performance is to be considered in the design of such distributed simulation systems, performance modelling must be employed to see that the benefits of distributed processing are achieved [2]. Since the structure of distributed simulators is not unique, we need tools to evaluate the performance of such simulators. Although analytic and simulation models have been recognized as useful tools, each tool has its distinct costs and advantages. Dubois [7] tried to combine such tools in complementary ways to exploit the advantages of the techniques. Nicol and Reynolds [11] built a network simulation model to observe parallelism and communication dependency within the components of distributed systems. The observations lead to a partitioning of the model which reduces the total cost of event-list maintenance and to make sure that the simulation work is distributed.

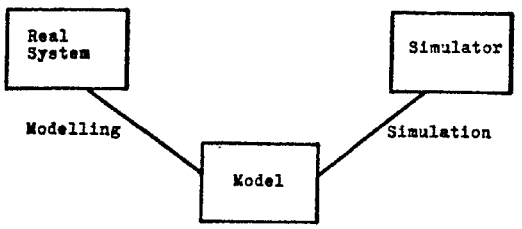
In order to compare relative performance of simulators, performance measures must be developed. Several research works indicated that the influential performance factors are workload balance between the cooperating processors and frequency of communications between processors [6]. Frankline [8] proposed a performance measure derived from analyzing the simulation cost that includes factors relating to the simulation machine speed, machine cost, waiting-time cost, and simulation quality. Recently Livny [9] has studied the relationship between the inherent parallelism of a concurrent simulation and the number of processors employed. He used as a performance measure the parallelism factor of the computation which is the ratio between the total processing time and its execution time. Nicol and Reynolds [10] claimed that "...excessive communication can lead to degraded performance, but minimizing communication need not optimize performance...". To integrate these various approaches and results a coherent methodology for performance evaluation of distributed simulators is needed.

In this paper, we describe a framework for performance modelling and simulation, and propose a performance simulator architecture called the *hierarchical multiport simulator* to evaluate performances of distributed simulation systems. We also design a simulation system based on the proposed methodology and the architecture. Using the simulation system, empirical results are collected and analyzed to find the optimal decomposition levels of several distributed simulator models. Finally, conclusions concerning the application of the performance evaluation methodology are given. A more detailed discussion can be found in Baik [1].

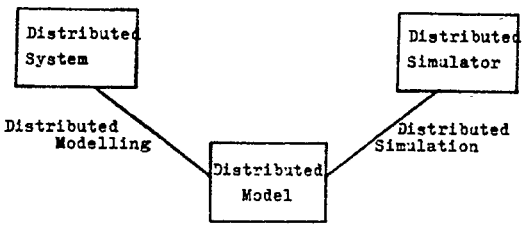
FRAMEWORK FOR PERFORMANCE MODELLING AND SIMULATION

Modelling and simulation is a set of activities which relates to constructing models of real world systems and simulating them on computer systems [14]. Based on this definition, general system modelling and simulation is concerned with three major entities - real systems, models, and simulators - and their relationships - modelling relation (between real systems and models) and simulation relation (between models and simulators). The *real system* is a source of input-output data which we obtain by input-output observation. That is, we are concerned with input-output relation of the real system. The *model*, a representation of the real system, is also a source of such data. The input-output relation of the model is obtained by experimentation with the *simulator*. Applying this entities-relationships to distributed systems, we can build an entities-relationships for distributed system modelling and simulation. Our interest is on performance evaluation of distributed simulator systems, we propose the entities-relationships for distributed simulator performance modelling and simulation as depicted in Figure 1. The

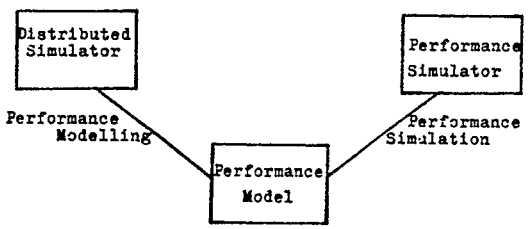
[1] This research was supported by the NSF grant DCR 8407230, "Distributed Simulation of Hierarchical, Multilevel Models".



(a) General System Modelling and Simulation



(b) Distributed System Modelling and Simulation



(c) Distributed Simulator Performance Modelling and Simulation

Figure 1. Entities-Relations of Modelling and Simulation

performance model is a representation of the distributed simulator with consideration of performance evaluation. The input-output relation of the performance model is obtained by experimentation with the *performance simulator*, which is a simulation processor capable of generating input-output data. There are five steps and six corresponding objects in the methodology for performance modelling and simulation. The five steps are namely conversion, transformation/mapping, implementation, simulation, and analysis, and the six objects are namely distributed simulator model (DSM), composition tree representation (CTR), hierarchical multiport simulator (HMS), performance simulator (PS), performance measures (PM), and performance results (PR). The methodology proceeds as follows:

1) Converting a given DSM into a tree structural representation called CTR.
2) Transforming the CTR to another CTR using operations soon to be defined and then mapping the transformed CTR onto a hierarchical performance simulator called HMS.
3) Implementation based on the HMS - selecting input parameters and initializing data structures such as the routing set.
4) Simulation with the PS to get performance measures.
5) Analysis of the PM with certain criteria such as speed or complexity.

Using the methodology, we propose a system called PMSS for performance modelling and simulation of hierarchically distributed simulators, as shown in Figure 2. It comprises two subsystems, the performance modelling subsystem (PMS) and the performance simulation subsystem (PSS). PMS is elaborated in Figure 3. It converts a given distributed system model into a composition tree model which is a tree structural representation of the model. PMS then maps the (transformed) tree model onto a hierarchical performance simulator which results from the modular construction of component modules to several levels of recursion.

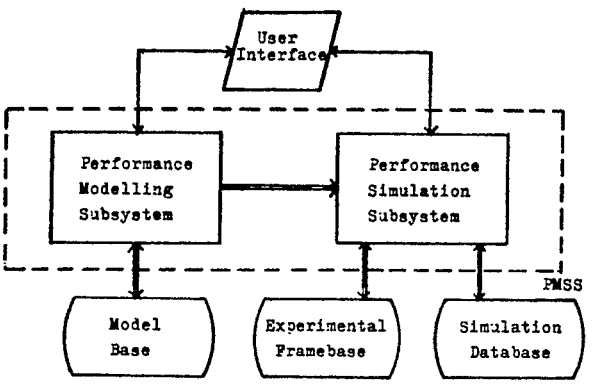


Figure 2. Performance Modelling and Simulation System (PMSS)

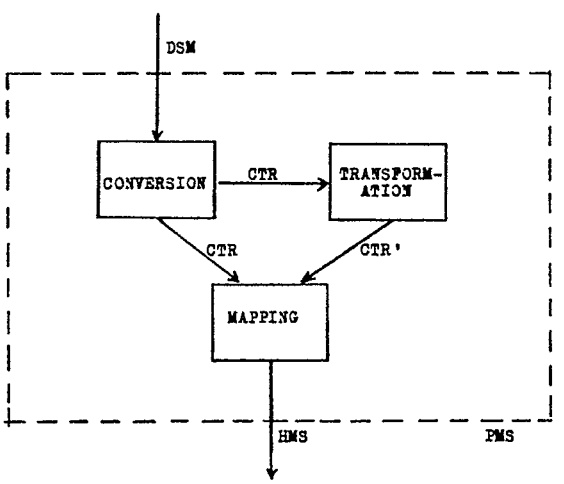


Figure 3. Performance Modelling Subsystem (PMS)

PSS depicted in Figure 4 comprises three procedures - implementation, simulation, and analysis. Based on the given hierarchical multiport simulator model, PSS implements a simulator by selecting parameters for delays and completing data structures including a coupling table and a routing set. And PSS simulates the model with the simulator in order to produce simulation outputs such as performance measures, and then analyzes them.

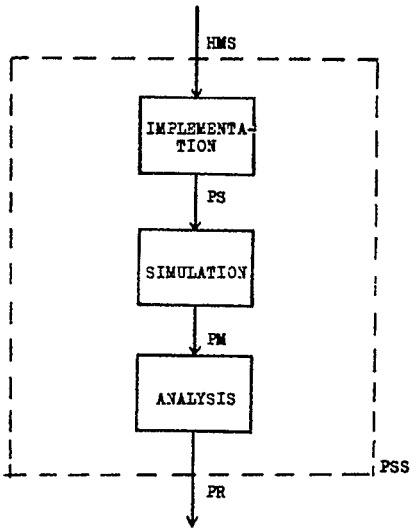


Figure 4. Performance Simulation Subsystem (PSS)

CONVERSION, TRANSFORMATION AND MAPPING PROCEDURES

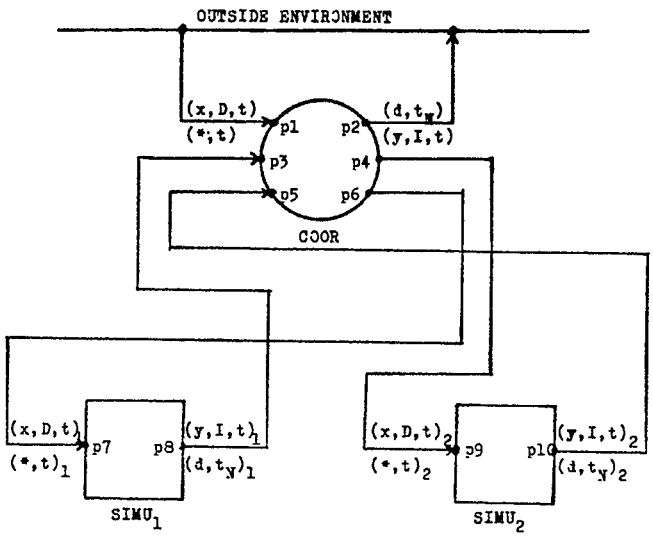
The distributed simulator model (DSM) proposed in this study is composed of the components representing those of the real system, and the couplings representing interactions among the components [4]. The composition tree representation (CTR) consists of interior nodes representing the compositions and leaf nodes representing the components of the distributed models. The hierarchical multiport simulator (HMS) contains simulator assumed to correctly simulate the corresponding model components, and coordinators responsible for synchronizing the component simulators and mediating their intercommunication. Procedures from a given DSM to a desired CTR employs two steps. The first step is the construction of a base CTR by node assignments and representation of the hierarchically specified DSM. The second step is the transformation of the base CTR to another structure of CTR by any of the following operations:

1) AGGREGATION - a many-to-one composition mapping from a base configuration satisfying the sufficient conditions to a transformed configuration constructed by block composition.
2) FLATTENING - a reconfigurable composition mapping, in which an interior node is removed by connecting directly its children nodes to its parent node.
3) DEEPENING - a reconfigurable composition mapping, in which the number of branches of an interior node is reduced by combining at least two branches, but not all, and adding one new interior node for the combined branches.

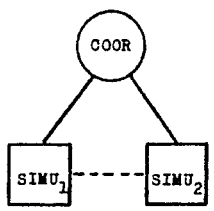
Therefore, aggregation can take place to any deepened or flattened composition in order to get more transformed configurations. Mapping a CTR onto a HMS is a one-to-one matching, that is, each of interior nodes of the CTR will match to a coordinator processor, and each of leaf nodes will match to a simulator processor. The hierarchy provides a formal way to manipulate models by using essential concepts such as association and morphisms [15]. These concepts are required to transform system specification from one form to another and to prove the preservation of structural features.

HIERARCHICAL MULTIPORT SIMULATORS

The hierarchical multiport simulator contains coordinators to synchronize the component simulators and handle tasks, and simulators to simulate the corresponding components. As shown in Figure 5, COOR has three input ports and three output ports. An input and output port pair such as (p1,p2) is for communicating with either its parent or the outside environment, and two pairs such as (p3,p4) and (p5,p6) are for communicating with $SIMU_1$ and $SIMU_2$. An input port (p7 or p9) of $SIMU_i$ is connected to an output port (p4 or p6) of COOR, while an output port (p8 or p10) of a $SIMU_i$ is connected to an input port (p3 or p5) of COOR.



(a) 1-level/2-fold HMS Architecture



(b) CTR of 1-level/2-fold HMS

Figure 5. 1-level/2-fold Hierarchical Multiport Simulator

There are two types of tasks for intercommunication, (x,D,t) and (*,t), and one for synchronization, (d,t_N). The task (y,I,t) contains the output event from either a simulator or a coordinator which is sent to the next level coordinator. Both coordinators and simulators carry out a set of subtasks for a given task such as (x,D,t), (*,t), (y,I,t), or (d,t_N),

where x is an external input event,
* is an internal input event,
y is an output event,
d is a done notice,
I is a set of Influencees,
D is a set of destinations,
t is a current time,
t_N is a time to the next event.

We now show how the different tasks are sent and received in a 2-level/2-fold hierarchical multiport simulator as an example. The arrival of (x,D,t) at COOR causes COOR to send $(x,D,t)_1$ and $(x,D,t)_2$ to $COOR_1$ and $COOR_2$ respectively, assuming that this task is for all simulators. Then $COOR_1$ sends $(x,D,t)_3$ and $(x,D,t)_4$ to its simulators, $SIMU_1$ and $SIMU_2$. This is similarly done by $COOR_2$. When a (*,t) arrives at COOR, a (*,t) is sent to the imminent child who has the minimum t_N. Supposing that $COOR_2$ and $SIMU_3$ are the imminent components, $COOR_2$ sends a (*,t) to $SIMU_3$. After receiving the (*,t), $SIMU_3$ immediately sends a (y,I,t) to its respective coordinator, $COOR_2$, and starts computation. When the (y,I,t) arrives at $COOR_2$, a (y,I,t) is sent to the next level coordinator, COOR, and it also sends a $(x,D,t)_7$ to $SIMU_4$, assuming that $SIMU_4$ is an influencee of the influencer, $SIMU_3$. This is then done by COOR in the same way. When all of (d,t_{Ni}) arrive at a coordinator, it determines the minimum, t_N, out of all t_{Ni}. The coordinator then sends (d,t_N) to either its parent or the outside if it is the root.

It should be noted that parallelism is achieved in the receipt of (x,D,t) and (*,t) tasks, because a (x,D,t) task can achieve parallelism when there are more than one destinations (i.e., $|D| > 1$), and a (*,t) task can achieve parallelism through the size of influencees of the influencer. A simulator aggregated with enclosed processors is called a *aggregated node simulator* and is assumed to be mapped on a sequential uniprocessor. The task execution of a (x,D,t) in the aggregated node simulator is different from that in a comparable non-aggregated node simulator, that is, when (x,D,t) arrives at the aggregated node simulator, (x,D,t) is sent to each enclosed simulator of the aggregated node simulator one at a time in a sequential mode. Therefore it is true that an aggregated node simulator is measured no parallelism but less hardware complexity against a comparable non-aggregated node simulator.

DEFINITIONS AND ASSUMPTIONS

We assume that a performance simulation model contains identical processors that communicate by message passing. Furthermore, we assume that the delay for communication, coordination, or computation is constant. And let,

- .complexity of a coordinator is 1,
- .complexity of a non-aggregated node simulator is 1,
- .complexity of an aggregated node simulator is the number of enclosed processors,
- .complexity of a given model is the summation of complexities of all processors plus the number of links among the processors.

Thus, the hardware complexity of a particular simulator model can be computed by these definitions. For example, the complexity of the fully decomposed 2-level/3-fold simulator is 25, and the complexity of the one level decomposed 2-level/3-fold simulator is 16. Here, the 2-level/3-fold simulator is a balanced tree having the maximum level of 2 and the number of branches of 3 to every interior node.

If the complexity measurement for hardware units is given, we can compute throughput per hardware unit of a particular simulator model. Comparing performance simulation outputs for each simulator model mapped from a given distributed model, an optimal model decomposition level can be found. Each model has an optimal decomposition level in terms of performance such as the minimum average of run.times or the maximum throughput per unit of hardware complexity. Here, the *run.time* is defined as the flow time of a task between the time the task gets into a simulator model and the time it gets out of the model. The *throughput* per unit of hardware complexity is defined as one over the product of the average run.time and the hardware complexity of the given model. Thus, to find the maximum throughput is to find the minimum product of the average run.time and the hardware complexity.

DESIGN OF PERFORMANCE SIMULATOR

Based on the methodology for performance modelling and simulation of hierarchical simulators described in the previous sections, we design a simulation system called the *performance simulator* to determine the performance of a given distributed simulator models, so that a performance simulation program is written in SIMSCRIPT II.5 [1]. A coordinator in a simulation model needs time for coordination (called COORDINATION.TIME) for table look-ups or to select a minimum t_N. A simulator on the other hand spends time to compute its next event time (called COMPUTATION.TIME). And messages which carry task information between processors require time for communication (called COMMUNICATION.TIME). To evaluate the performance of a given distributed simulator model in terms of time delays, the simulation model parameters: COORDINATION.TIME, COMPUTATION.TIME, and COMMUNICATION.TIME, and the experimental frame parameters such as an OBSERVATION.INTERVAL are given. A simulation system carries out simulations with these parameters and then output results in statistical summaries such as RUN.TIME.

The performance simulator is comprised of the simulation model and the experimental frame as shown in Figure 6. The simulation model contains a set of coordinator-processes and a set of simulator-processes coupled to each other, and two types of globally accessable data structures. These tables are the coupling table and the routing set, which are used by all processors, coordinators and simulators, for routing and searching purposes:

1) COUPLING.TABLE - contains information on parent-children couplings and influencer-influencee couplings. This table is built from given input data on the hierarchical simulator to be simulated for evaluating its performance.
2) ROUTING.SET - contains routing information on the routes from each simulator to the root. Whenever a task comes into a coordinator, the ROUTING.SET is searched to find a route for sending tasks down to the next level. This table can be built from the coupling table.

Employing the DEVS formalism, the structrual representation of an experimental frame is defined as a coupling of a generator, an acceptor and a transducer. The experimental frame specifies three systems connected to the model as shown in Figure 6. The generator is an input system of the input segments S_I. The transducer is an output system which observes model input/output segment pairs and performs the statistical processing specified by the summary

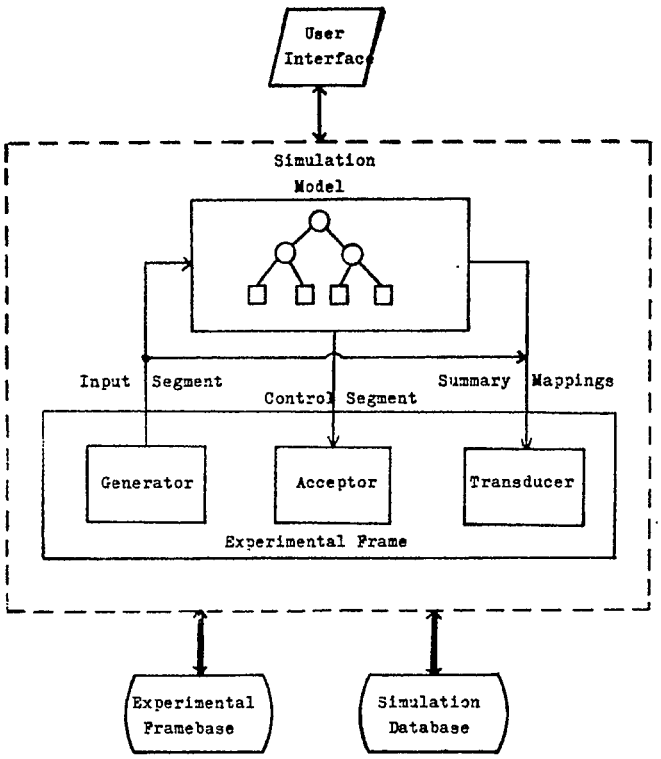


Figure 6. Performance Simulator

mappings SM. The acceptor is a run control system which observes a run control variable segment and indicates acceptance or rejection of an experiment according to whether or not the segment belongs to the admissable class S_C. For performance evaluation, the generator creates tasks such as input events to be sent to the simulation model, the acceptor controls the simulation run by checking current simulation time against the given observation interval, and the transducer performs statistical computation such as average run-time per task.

Process interaction oriented simulation languages such as SIMSCRIPT [13] can be used to implement the performance simulator. Each component of a distributed simulator model will be implemented by a process and messages are exchange for synchronization and communication.

EXPERIMENTS AND RESULTS

Once the performance simulator is established, we can examine several configurations of distributed simulator models with constraints such as the number of processors or the number of links among the processors. By using the performance simulator, we can evaluate the performances of different configurations for a given distributed simulator model in order to find an optimal one. We therefore can decide at what level to terminate the recursion in the hierarchical model specification so that the coupling of systems associated with that level satisfies the constraints of the simulator model. To get empirical results by using the performance simulator, we set up the input parameters as follows:

COMPUTATION.TIME = 8.0 time units
COORDINATION.TIME = 0.5 time units per branch
COMMUNICATION.TIME = 1.5 time units

Figure 7 shows the performance simulation results of 2-level/k-fold (k=2,3,4,5) hierarchical multiport simulator model. While the fully decomposed level of a 2-level/k-fold simulator model is optimal in terms of the minimum average run.time, the one level decomposed simulator model is optimal in terms of the maximum throughput per unit of hardware complexity. As shown in Figure 8, while the two level decomposed model of a 3-level/2-fold simulator is optimal in terms of the minimum average run.time, the one level decomposed model is optimal in terms of the maximum throughput per unit of hardware complexity. However, in the 4-level/2-fold simulator, the three level decomposed model is optimal in both cases.

When the number of destinations, $|D|$, of the external tasks, (x,D,t), and the number of influencees, $|I|$, of the output tasks, (y,I,t), are considered, the following four combinations are chosen:

1) Tight-inner-coupling/Tight-outer-coupling (Tight/Tight)
2) Tight-inner-coupling/Loose-outer-coupling (Tight/Loose)
3) Loose-inner-coupling/Tight-outer-coupling (Loose/Tight)
4) Loose-inner-coupling/Loose-outer-coupling (Loose/Loose)

Here, the *inner-coupling* means the internal influencer-influencee relationships of the model, and

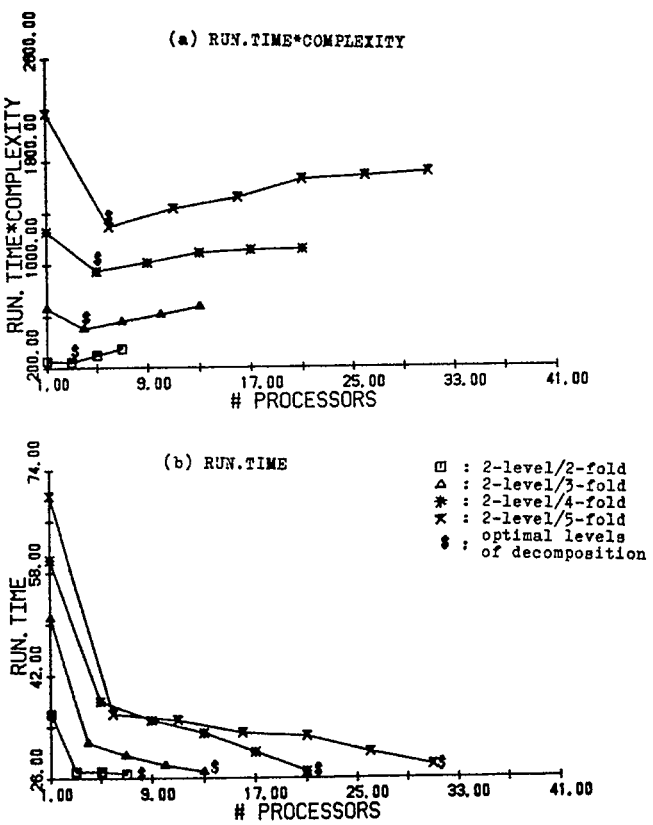


Figure 7 . Simulation Results of 2-level/k-fold Simulators

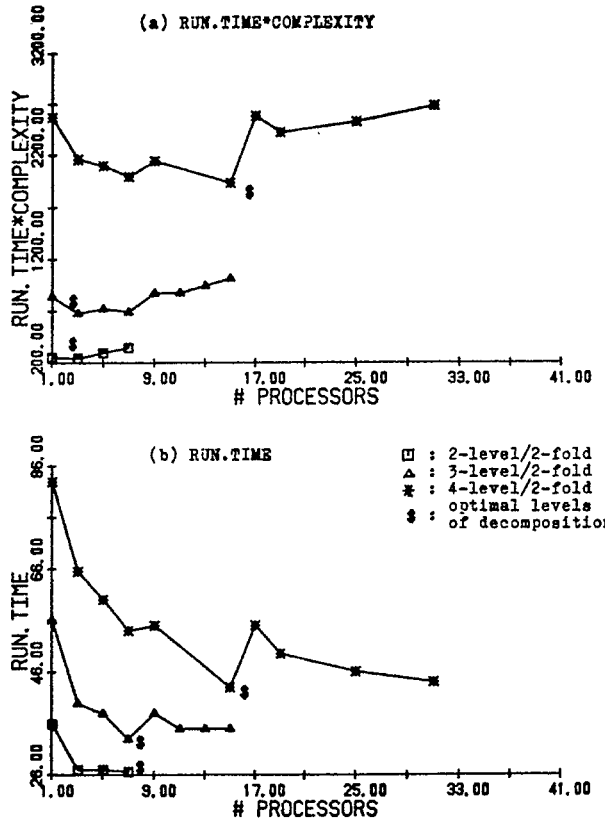


Figure 8. Simulation Results of k-level/2-fold Simulator

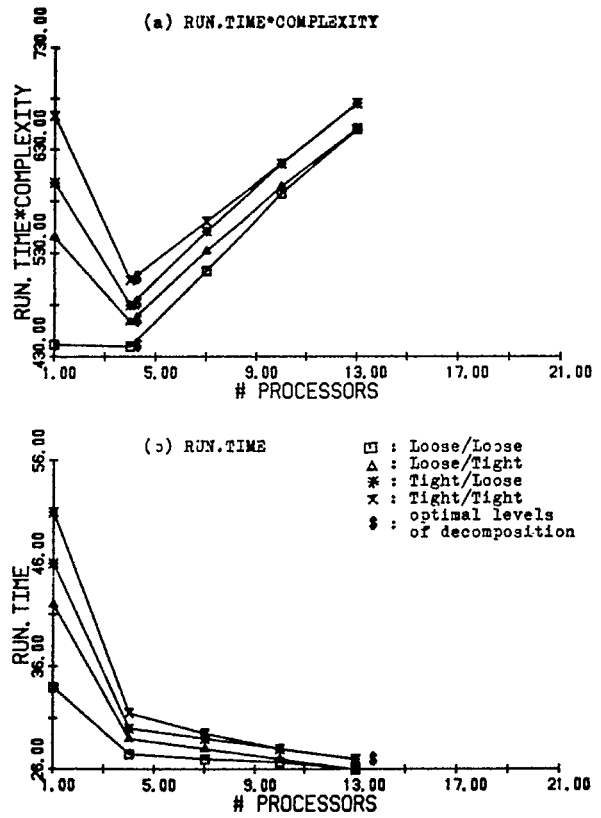


Figure 9. Simulation Results of 2-level/3-fold Simulator with consideration of coupling

the *outer-coupling* means the external relationships to the environment. And the *tight-outer* means that the number of destination, $|D|$, is relatively bigger than that of the *loose-outer*. For instance, in our expriments, the tight-outer is assumed that $|D|$ is greater than half of the number of simulator processors, and the loose-outer is assumed that $|D|$ is not greater than half. However, determining whether the simulator model is the *tight-inner* or the *loose-inner* depends on the number of influencees, $|I|$, of each leaf node.

In the experiments shown in Figures 7 and 8, only the first combination, the tight-inner-coupling and the tight-outer-coupling, are given by default. In the 2-level/3-fold simulator model, experimental results show that two optimal performance levels are found, one is the fully decomposed level in terms of the minimum average run.time, and the other is the one level decomposition in terms of the maximum throughput per unit of hardware complexity. See Figure 9. Each level is run with the following coupling relationships:

1) Tight/Tight is the case where $|I| \geq 4$ and $|D| \geq 4.5$
2) Tight/Loose is the case where $|I| \geq 4$ and $|D| < 4.5$
3) Loose/Tight is the case where $|I| < 4$ and $|D| \geq 4.5$
4) Loose/Loose is the case where $|I| < 4$ and $|D| < 4.5$

CONCLUSIONS

Several architectures were developed for hierarchically specified distributed simulators based on the DEVS formalism. This has been done by mapping hierarchical modular discrete event models onto distributed architectures [15]. Zeigler introduced a concept called the *abstract simulator* which is an intermediate logic form in realizing the distributed model on a distributed simulator. Based on the concepts of the extended DEVS formalism, Concepcion [4] proposed an architecture for distributed simulation, the Hierarchical Multi-bus Multiprocessor Architecture, HM^2A. This architecture allocates processing elements to components of the abstract simulator. The Heterogeneous Element Processor, HEP, offers an alternative implementation for the realization of such distributed simulators [15]. Rozenblit [12] presented the abstract simulator of distributed models with experimental frames by mapping hierarchical specification of experimental frames onto the abstract simulator. Since these architectures are, however, fixed in how they do the simulation of the distributed models, the model decomposition is necessary to be chosen for evaluating performance of such architectures.

The analysis of performance for the proposed architectures with the constraints such as the number of processors or the number of links among the processors, are required to study the feasibility of

such hierarchical simulators. This can be done by simulating such architectures to determine the performance of different decomposition levels of distributed simulators. Therefore the major concern, in this study, is to find the optimal decomposition levels in terms of performance criteria such as speed-up or parallelism. This is a very important aspect of using DEVS formalisms in a distributed simulation environment.

In this paper, we proposed (1) a methodology for performance modelling and simulation of distributed simulators, (2) a simulator architecture called the "Hierarchical Multiport Simulator (HMS)", which is an abstract simulator of distributed simulators, and (3) a technique to design a simulation system to determine the performance based on experimental aspects. Additionally, our experiments indicate that simulation approaches are suited to find the optimal decomposition levels of given distributed simulator models. This is done by comparing performance measures such as the AVG.RUN.TIME or the product of the AVG.RUN.TIME and the COMPLEXITY.

REFERENCES

1. Baik, D.K. and Zeigler, B.P., "Performance Modelling and Simulation of Hierarchical Distributed Simulators", Technical Report CSC-85-003, Department of Computer Science, Wayne State University, Detroit, Michigan, May 1985.

2. Bennett, D.A., "Simulation of A Distributed System for Performance Modelling", ACM Performance Evaluation Review, Vol.8, No.3, Fall 1979.

3. Bryant, R.E., "Simulation on A Distributed System", IEEE Proceedings of Distributed Computing Systems, 1979.

4. Concepcion, A.I., " Distributed Simulation on Multiprocessors: Specification, Design and Architecture", PhD Dissertation, Wayne State University, 1984.

5. Concepcion, A.I., Baik, D.K. and Zeigler, B.P., "Distributed Simulation of Hierarchical Models", Proc. of the Workshop on Parallel Processing using the HEP, Norman, Oklahoma, March 1985.

6. Davidson, D.L. and Reynolds, P.F., "Performance Analysis of a Distributed Simulation Algorithm based on Active Logical Processes", Proc. of the Winter Simulation Conf., 1983.

7. DuBois, D.F., "A Hierarchical Modeling System for Computer Network", ACM Performance Evaluation Review, Vol.11, No.1, Spring 1982.

8. Franklin, M.A., Wann, D.F. and Wong, K.F., "Parallel Machines and Algorithms for Discrete-Event Simulation", Proc. of Parallel Processing Conf., IEEE, 1984.

9. Livny, M., "A Study of Parallelism in Distributed Simulation", Proc. of the Conf. on Distributed Simulation, San Diego, Jan. 1985.

10. Nicol, D.M. and Reynolds, P.F., "Problem Oriented Protocol Design", Proc. of the Winter Simulation Conf., 1984.

11. Nicol, D.M. and Reynolds, P.F., "A Statistical Approach to Dynamic Partioning", Proc. of the Conf. on Distributed Simulation, San Diego, Jan. 1985.

12. Rozenblit, J.W., "Experimental Frames for Distributed Simulation Architectures", Proc. of Distributed Simulation, San Diego, Jan. 1985.

13. Russell, E.C., Building Simulation Models with SIMSCRIPT II.5, CACI, L.A., 1983.

14. Zeigler, B.P., Theory of Modelling and Simulation, Wiley, N.Y., 1976.

15. Zeigler, B.P., Multifacetted Modelling and Discrete Event Simulation, Academic Press, 1984.

DOO-KWON BAIK is a PhD candidate of Computer Science Dept. at Wayne State University. His research interests include modelling and simulation, distributed systems, performance evaluation, and software engineering. He received BS in Mathematics and MS in Industrial Engineering from Korea University in 1974 and 1977, and MS in Computer Science from Wayne State University in 1983. He is a member of ACM, IEEE, SCS, and ORSA.

Doo-Kwon Baik
Dept. of Computer Science
Wayne State University
Detroit, MI 48202
(313)577-2477

BERNARD P. ZEIGLER is a professor of Electrical and Computer Engineering Dept. at University of Arizona. He is author of "Multifacetted Modelling and Discrete Event Simulation", Academic Press, 1984, and "Theory of Modelling and Simulation", Wiley, 1976. His interests include distributed simulation and expert systems for simulation methodology.

Bernard P. Zeigler
Dept. of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona 85721
(602) 621-2434