Distributed Simulation over Network of Workstations

Hristo Valchanov, Nadezhda Ruskova, Trifon Ruskov

Abstract: *Parallel discrete event simulation (PDES) is a basic approach for evaluation of the complex systems. A PDES attempts to speed up the execution of a simulation by distributing the simulation's workload between multiple processors. A network of workstations is a widely available platform for PDES. The present article provides a comparative analysis of operational distributed simulation models based on the TimeWarp algorithm. An operational model allowing for reduction of the simulation costs has been proposed based on an experimental evaluation.*

Key words: *Distributed Simulation, TimeWarp, PDES.*

INTRODUCTION

PDES allows for acceleration of the modeling process by distributing it among a number of processors. With PDES, the modeled system is presented as a set of sub-systems simulated by a number of simulation objects (SO) communicating with one another by means of exchange of timestamped messages for occurring events. The simulation correctness requires that the events be processed in the order of their occurrence in time. Special synchronizing protocols are used to ensure the right order of processing. One of the most popular synchronizing protocols is the TimeWarp protocol [1].

Using the TimeWarp protocol, the so-called optimistic approach to a simulation is implemented, where the simulation objects process the events by the order of their receiving. After each event has been processed, the respective SO moves its local time to the time of the event. The order of events received, however, may differ from the order of occurrence in time producing a causality error. This error occurs upon receipt of a message with timestamp less than the current local time of the simulation object. Such a message is known as a *straggler*. To eliminate the error, SO has to restore the state as it was before the strangler message had been received and cancel all messages sent by it after that moment. For that purpose the simulation objects have to keep information on their state on a regular basis. Cancellation of a sent message is effected by sending the respective anti-message. Restoring of the state is called *rollback* of the simulation.

The simulation objects may be implemented as separate independent processes. Such implementation, however, is ineffective from the point of view of the high system overhead on switching of the processes context by the operating system (OS). On the other hand, the communications between the processes in a same workstation is implemented by the OS IPC messages mechanism and has approximate complexity as the intercomputer network communications. By these reason the speed of simulation is largely reduced. Simulation effectiveness improvement can be achieved by aggregating the simulation objects into a cluster. Each cluster will perform as an independent process within a workstation. Its purpose is to carry out scheduling of its simulation objects and ensure communication with the other clusters within the network.

The present article deals with certain operational models for implementing of simulation clusters and provides experimental evaluations of their effectiveness. Various parameters of the simulation process have been studied and analyzed based on which an operational model of distributed simulation in a network of workstations allowing for performance improvement has been defined. A local network of personal computers connected by 100Mb Ethernet running under the Linux has been used as a platform. The communication between the clusters has been realized by means of the MPI message passing library [2].

OPERATIONAL MODELS OF A SIMULATION CLUSTER

An important feature of the distributed simulation is the so-called granularity. Granularity is defined as a ratio between the individual event performance time and the simulation total time. In the greater part of PDES systems described in the literature, each component of the real modeled system is presented in its simulation model as a separate simulation object. In practice, the direct real/simulation object mapping may result in events of relatively low granularity, that is, the events performance time is considerably shorter than the simulation system maintenance time. Presentation of several modeled components with a single simulation object may increase the simulation granularity thus allowing for the internal communications in the simulation model to be implemented by means of a common memory. Within the simulation cluster thus formed, the events can be processed by sequental simulation, with optimistic TimeWarp based synchronization used between the clusters.

Fig. 1 shows an operational model of a simulation cluster used in the WARPED simulation kernel [3]. It includes a communication and a simulation module. Both modules are implemented as threads running in parallel. This operational model is used as a basic model in the present article.

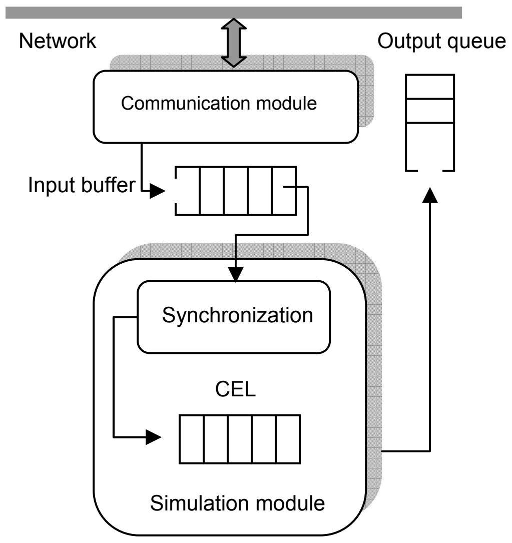


Fig. 1 The WARPED operational model

The simulation module functions on the basis of a cluster event list (CEL) ordered by times of occurrence. The events ordering in the list ensures correctness of the simulation in the cluster. The module performs two main operations:

- Synchronization of the events;
- Execution of the code of the simulation objects modeling the occurrence of those events.

The messages generated as a result of the processing for SO in the same cluster are directly inserted in the CEL. The messages designated for other clusters are deposited in the output queue at the communication module.

The communication module carries out the communication with the simulation clusters being executed at the various workstations. It ensures both receipt of the messages of occurring events from other clusters and transmission of the messages from the output queue.

It is typical of the operational model described that the synchronization operations are executed in the code of the simulation module where messages synchronization and processing are sequential operations. The module first checks for new messages in the input buffer and executes the synchronization algorithm. In case a straggler message is found, the causality error is eliminated by launching the actions for restoring the previous correct state. Any message of normal time order will be deposited in the CEL. Then the simulation module continues with the processing of the events of the CEL.

Using this operational model for the simulation on architectures with distributed memory may lead to unfavorable results manifested in decrease of the simulation speed. Due to the parallel execution of the simulation objects, the asynchronous character of the exchange of message between them and the delays typical of the network communications, the probability of a straggler message receipt is extremely high. The parallel functioning of the modules, on the other hand, presumes for the messages receipt to be independent of the events processing. Thus, in case of a sequence of several straggler- or anti-messages, they will be processed in separate synchronization cycles of the simulation module. In each cycle the operations of anti-messages sending and restoration of the previous correct state will be repeated. As a consequence thereof the following negative results may be arrived at:

- growth in the number of anti-messages;
- increase of the costs on restoring the state;
- repeated messages processing and transmission.

A possible solution for reduction of the above costs is integrating the functions of the communication and simulation modules in a single component. This approach is used in the implementation of GTW [4]. With this approach each simulation cycle first checks for the presence of a message from the communication subsystem and performs the synchronization operations, then processes the event and finally sends the generated messages to the communication subsystem for delivery. However, GTW has been designed for computing systems with shared memory only.

In the case of platforms with distributed memory, as the networks of workstations are, communication is implemented by means of message passing libraries. The semantics of the MPI library used presupposes performance of each *send* operation upon execution of its corresponding *receive* one. This feature does not allow for the simulation objects to transmit faster and more messages than the recipients can process. As a result, the restorations of the previous correct state will occur more rarely, which will lead to better time indexes of the simulation.

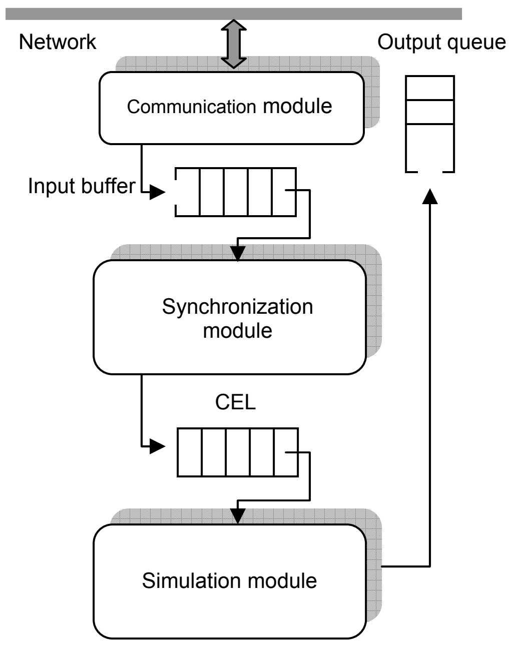


Fig. 2 The SYNCH operational model

However, the GWT operational model has a major fault – it cannot be used for dynamic creation of simulation objects. The semantics of the simulation of dynamic discrete systems requires blocking of the object-creator till starting-up of the newly created object and the return of its identifier. The creator will be blocked when a new object is created in another cluster. This prevents receiving of confirmation for the object creation. As a result, a situation of deadlock of simulation will arise.

In the present article we suggest an operational model (called SYNCH) by means of which the faults of the models discussed can be eliminated. The model consists of three modules: communication module, simulation module and synchronization module (fig. 2), implemented as threads running in parallel within a single simulation cluster.

The purpose of the communication module is similar to using it in the basic operational model. The important difference lies in the separation of the synchronization operations from the simulation module into an independent synchronization module. Upon completion of the synchronization, the events will be deposited by the synchronization module in the cluster list of events (CEL), to be next successively processed by the simulation module.

Unlike the basic model where the events synchronization and processing phases follow in sequential, they are executed simultaneously in the proposed model. In the event of rollback, the messages generated at the end of the processing phase will not be sent and the simulation process will continue with the cluster list of events restored in correct state. Whereas with the basic model the simulation objects are allowed to develop further in the model time between two rollbacks, this possibility is suppressed in the proposed model. In this way the rollback depth is limited to a smaller scope. The following results thereof are expected:

- decrease of the number of anti-messages sent within the network;
- reduction of the costs on restoring the state;
- reduction of the number of repeatedly sent normal messages.

EXPERIMENTAL EVALUATION AND ANALYSIS

Two benchmarks have been used for the experimental evaluation of the operational models described above. PHOLD is a popular simulation model defined by Fudjimoto (fig.3a) [5]. The models include N objects connected into a 2D torus network and E events circulating between the objects. Each object processes an event and generates an event to each of its four neighbors whereby its timestamp grows by a random value with exponential distribution. The second model is a model of a closed computer network with priority queues including N routers and Q queues connected to each of them (fig.3b) [6]. The routers generate E events to a queue selected by means of a normal distribution. The growth of the generated events time is subject to an exponential distribution.

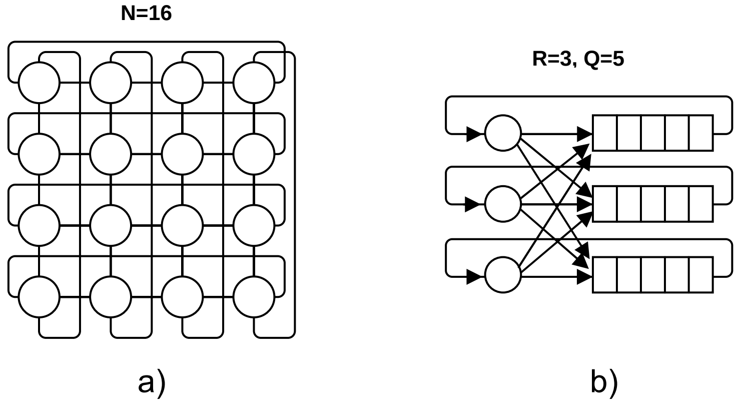
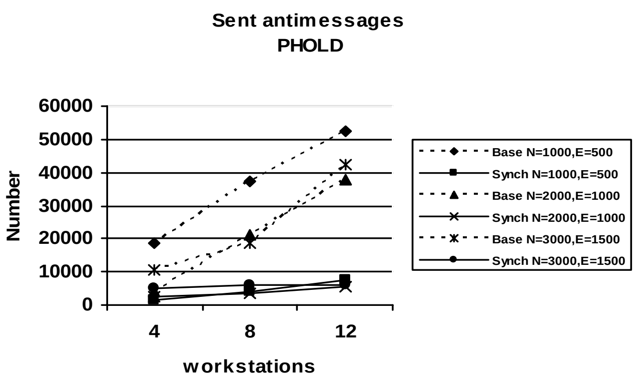


Fig. 3 Experimental models

These models are often used for evaluation of distributed simulation systems. They differ by the manner of mapping of the simulation models to the processor elements, by the number of circulating events and by their sensitivity to rollback.

The experiments have been carried out on a local network of Linux-based 900 Mhz Pentium III computers with 256Mb RAM, connected via 100Mb Ethernet. The following simulation parameters have been studied: execution time, number of processed messages, number of sent normal messages, number of sent anti-messages. The experiments have been performed with various values of the number of simulated objects and events till reaching a specified value of the model time.

The diagrams on fig. 4 show a considerable decrease in the number of sent anti-messages with the operational model proposed by us. At the same time, it is noted that the increase of the number of anti-messages remains within narrow limits regardless of the growth of the simulation size (number of objects and events) as well as of the number of workstations. This is a prerequisite for cutting down the communication costs typical of increasing the number of workstations.



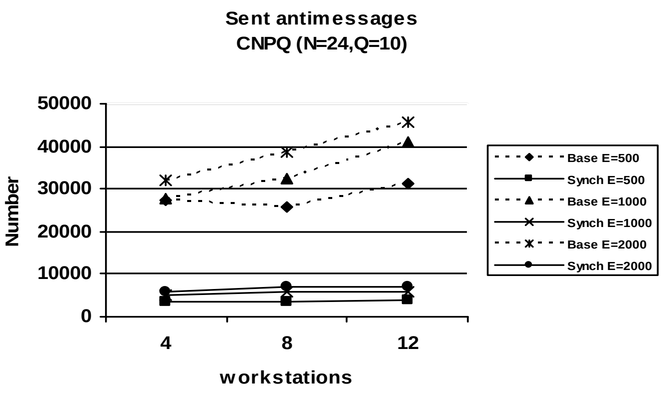
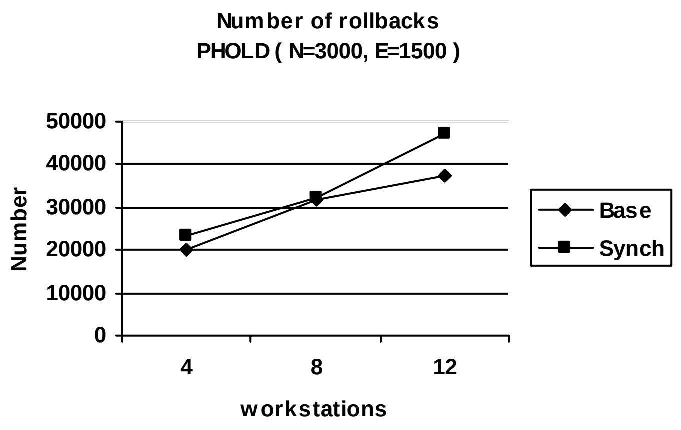


Fig. 4 Sent anti-messages charts

As a result of the independent operations performed by the synchronization module an increase of the number of rollbacks in the cluster is noted in the SYNCH operational model compared to the basic one (fig. 5). This results in an increase of the costs on restoring the previous correct state. However, as a whole it reduces the execution time as shown by the results presented on fig. 6.



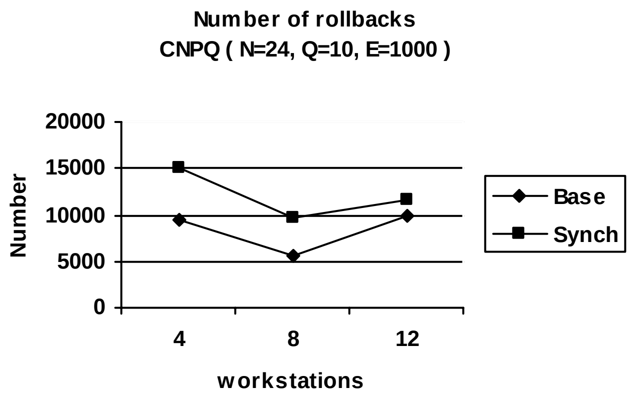


Fig. 5 Number of rollbacks charts

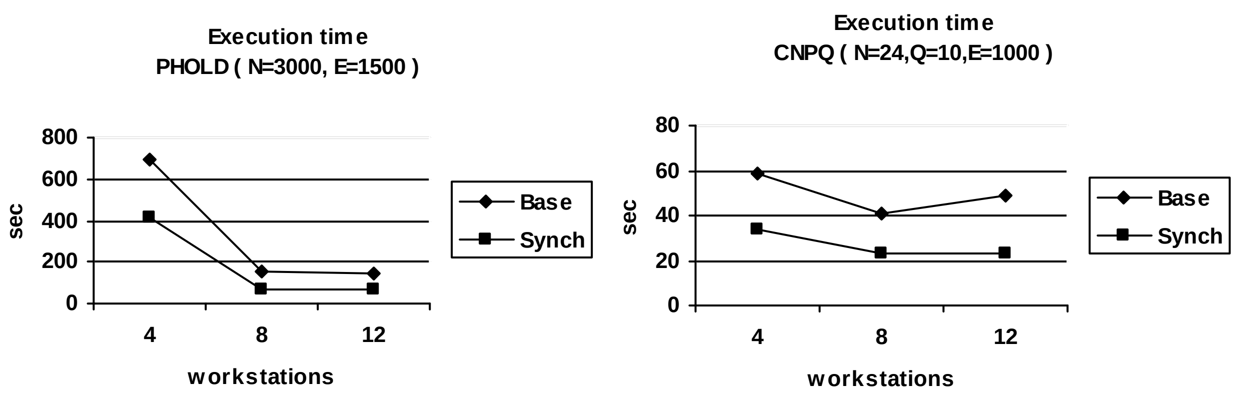


Fig. 6 Execution time

CONCLUSIONS AND FUTURE WORK

An operational model of a simulation cluster for distributed simulation in a network of workstations has been presented in the present article. Based on an experimental evaluation and a comparative analysis by means of a basic operational model, its higher effectiveness has been proved. The objective of our future work will be the improvement of the model in view of reducing the costs on restoring the previous correct states as well as of the anti-messages exchange between the individual clusters.

REFERENCES

[1] Fujimoto R.M. Parallel Discrete Event Simulation. Communications of the ACM, vol.33, N10, 1990, 41-52.

[2] MPI Forum, MPI2: A message-passing interface standard. The International Journal of High Performance Computing Applications 12(1-2), 1998, 1-299

[3] Dale E., Wilsey P., Timothy J. WARPED Simulation Kernel Documentation, 1995.

[4] S. Das, R. M. Fujimoto, K. Panesar, D. Allison and M. Hybinette. "GTW: A Time Warp System for Shared Memory Multiprocessors." In Proc. of the 1994 Winter Simulation Conference, 1332–1339.

[5] Fujimoto R. Performance of TimeWarp under synthetic workloads. Proc. Of the SCS, 1990, 23-28.

[6] Ruskova N., Walchanov H. Interprocess communication in distributed simulation systems. Proc. of the CompSysTech'2000, Sofia, 2000, I.9-1 - I.9-5.

ABOUT THE AUTHOR

Assist. Prof. Hristo Valchanov, Department of Computer Science and Engineering, Technical University of Varna, Bulgaria. Phone: +359 52 383424, E-mail: hristo@tu-varna.acad.bg

Assoc. Prof. Nadezhda Ruskova, PhD, Department of Computer Science and Engineering, Technical University of Varna, Bulgaria. Phone: +359 52 383424, E-mail: ruskova@tu-varna.acad.bg

Assoc. Prof. Trifon Ruskov, PhD, Department of Computer Science and Engineering, Technical University of Varna, Bulgaria. Phone: +359 52 383424, E-mail: ruskov@tu-varna.acad.bg