A Performance Study of Run-time Systems for Distributed Time Warp Simulation

Hristo G. Valchanov[1], Nadezhda S. Ruskova[2] and Trifon I. Ruskov[3]

Abstract - **Parallel discrete event simulation (PDES) is a basic approach for evaluation of 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 Time Warp algorithm. An operational model allowing for reduction of the simulation costs has been proposed based on an experimental evaluation.**

Keywords – **Distributed simulation, Time Warp, PDES.**

I. INTRODUCTION

PDES allows acceleration of the simulation process through its distribution among a number of processor. The modeled system is represented as a set of components, simulated through simulation objects (SO). SO exchange timestamped messages, simulating the events occurring in a simulation model. The correctness of simulation (processing of events in their correct order of occurrence) is guaranteed by special synchronization protocols, with Time Warp being one of the most widely used nowadays [1].

The Time Warp protocol allows SO to process the events according to the order of their receiving – the so called "optimistic approach". After processing of each event, SO moves up its own local time equal to the time of occurrence of the event. The approach allows processing of events in a sequence, different from that of their natural occurrence, which would result in a casualty error. Such an error occurs in case of receiving an event message (called *straggler*) with a timestamp lower than the current time of SO. The procedure of error correction requires SO to recover its state before the time of arrival of a straggler, and to cancel all messages sent up to that moment. For this purpose, each SO performs a periodic saving of its state. Canceling of all sent messages is effected through sending of corresponding anti-messages. The entire process of SO recovery is known as *rollback*.

An important feature of the discrete-event simulation is the so-called *granularity* [2]. Granularity is defined as the ratio between the individual event performance time and the simulation total time. The direct real-to-simulation object mapping can result in events with very small granularity, i.e. the time for processing of a single event may turn out to be much smaller than the system time, spent for its simulation. The representation of many modeled components in a single SO may increase simulation granularity thus allowing for the internal communications in the simulation model to be implemented by means of a common memory. Within the SO cluster, formed in this way, the events can processed by sequential simulation, based on a time-ordered event list – *cluster event list* (*CEL*), whereas, for the communication between the clusters, the Time Warp protocol is used.

Each simulation cluster consists of two main components – simulation (SM) and communication modules (CM), through which distribution of cluster functionality is achieved. Both components can work in parallel, e.g. as threads executed in parallel. The SM processes the time-ordered queue of events from CEL. The CM ensures the exchange of messages between the clusters, executed on different processors. Such structure is common for the kernels of a number of software simulation packages, as WARPED, PARSEC, etc [3], [4]. A typical feature of their operational models is that the synchronization is performed within the code of the simulation module, in which the actions of message synchronization and message processing are sequential operations.

The usage of this operational model (called *BASE*) for simulation on distributed memory architectures, like network of workstations, can lead to undesirable effects, resulting in decrease of simulation performance. Due to the parallel work of SO, the asynchronous message exchange between them and the delays, typical of the network communications, a high probability for receiving of a straggler message exists. On the other hand, the parallel functioning of the modules allows receipt of the messages to be independent from their synchronization and processing. Thus, in case of receipt of a sequence of straggler- or anti-messages, these will be processed in different synchronization cycles of SM, with each cycle repeating the actions for state recovery and canceling of sent messages.

In the present paper, the process of simulation state recovery is discussed, and an operational model for simulation performance improvement is proposed.

II. THE BASE ROLLBACK MODEL

The simulation process can be represented as a sequence of simulation cycles, each one composed of synchronization (S_i) and processing (P_i) phases. Processing of a message, assigned

[1]Hristo G. Valchanov is with the Computer Science & Engineering Department, Technical University of Varna, Bulgaria, E-mail: hristo@tu-varna.acad.bg

[2]Nadezhda S. Ruskova is with the Computer Science & Engineering Department, Technical University of Varna, Bulgaria, E-mail: ruskova@tu-varna.acad.bg

[3]Trifon I. Ruskov is with the Computer Science & Engineering Department, Technical University of Varna, Bulgaria, E-mail: ruskov@tu-varna.acad.bg

to SO, is performed without interruption, with checks for received stragglers and anti-messages being executed between two sequential event processing actions only.

The occurrence of a situation, in which a straggler or anti-message is received during processing phase P_i, will be defined as *violation of simulation correctness* (*VSC*). In the present Chapter, we shall formulate *simulation effectiveness* with respect to simulation process recovery in the event of *VSC*.

Fig.1 shows a situation, in which single straggler or anti-message is received during the processing phase of the i^{th} simulation cycle.

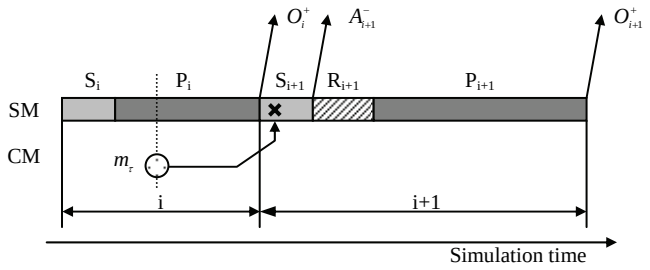

Fig.1. A single simulation correctness violation

Let m_i be a message for an event e_i, which is being processed in the i^{th} simulation cycle P_i. At the end of the P_i, the generated messages for the scheduled events within this cycle of the simulation - O_i^+, will be sent out. Let m_τ be a violator-message with a timestamp lower than the *cluster virtual time* $(\tau < CVT)$, received during the P_i phase. The actions for synchronization of this event will not be undertaken until the beginning of the next simulation cycle S_{i+1}. These will result in a rollback (R_{i+1}) to the correct state, corresponding to simulation time τ, and sending of canceling anti-messages:

$$A_{i+1}^- = O_i^- + O^-(k_{CVT,\tau}), \quad (1)$$

where O_i^- - number of anti-messages, necessary for canceling of the just sent normal O_i^+ messages;

$O^-(k_{CVT,\tau})$ - number of anti-messages, necessary for canceling of the normal messages sent out for all ***k*** events processed within the time interval between the current *CVT* and the violator-message timestamp τ. The number ***k*** defines the depth (in terms of number of events) of rollback for correct state recovery (R_{i+1}).

During the next simulation cycle P_{i+1}, all events in the recovered correct *CEL* (thanks to rollback R_{i+1}) will be processed. At the end of the cycle, normal messages will be sent again:

$$O_{i+1}^+ = O_i^+ + O^+(k_{CVT,\tau}) + O^+(\tau), \quad (2)$$

where O_i^+ - number of generated messages due to repeated event processing in the i^{th} simulation cycle;

$O^+(k_{CVT,\tau})$ - number of generated messages due to repeated event processing in the time interval (CVT, τ) due to rollback;

$O^+(\tau)$ - number of generated messages due to processing of message m_τ. (In case of m_τ is an anti-message, $O^+(\tau)$ =0).

In the example presented, the *VSC* correction is done during a subsequent simulation cycle. This causes performance overhead, related to sending of normal messages, followed by cancellation and repeated sending of the same. At the same time, the repeated processing of event e_i must be pointed out.

Effectiveness is further decreased in case of subsequent straggler and anti-messages, following one by one (Fig.2).

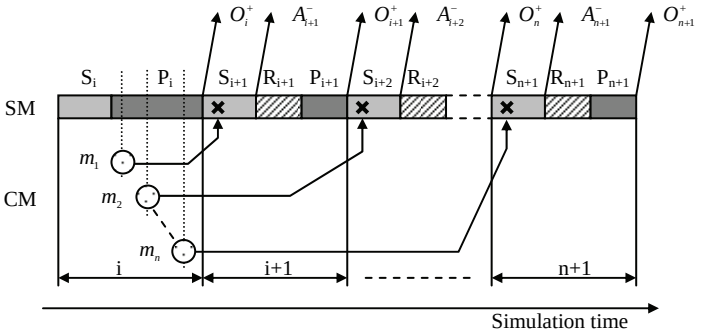

Fig.2. Multiple simulation correctness violation

Eqs. (1) and (2) can be derived for ***n*** violations, occurring in the i^{th} simulation cycle within a model time interval (τ, γ), where $\tau, \gamma < CVT$ (τ is the timestamp of the first message m_1 in the interval, γ is the timestamp of the last message m_n), as follows:

$$A_n^-(i) = n * A_{i+1}^- + \sum_{e=1}^{n-1} O_e^-(t_{\tau \le t \le \gamma}) + O^-(k_{\tau,\gamma}), \quad (3)$$

where $\sum_{e=1}^{n-1} O_e^-(t_{\tau \le t \le \gamma})$ - number of anti-messages, necessary for canceling of the normal messages sent after processing of the stragglers, included in *CEL* after state recovery;

$O^-(k_{\tau,\gamma})$ - number of anti-messages, necessary for canceling of the normal messages sent for all processed ***k*** events within the interval (τ, γ), in case of $\gamma < \tau$.

Similarly,

$$Q_n^+(i) = n * O_{i+1}^+ + \sum_{e=1}^{n} O_e^+(t_{\tau \le t \le \gamma}) + O^+(k_{\tau,\gamma}), \quad (4)$$

where $\sum_{e=1}^{n} O_e^+(t_{\tau \le t \le \gamma})$ is the number of generated messages due to the processing of violator-messages. If case of anti-messages, the value of this expression is zero;

$O^+(k_{\tau,\gamma})$ - number of messages generated by the repeated event processing within the interval (τ, γ), in case of $\gamma < \tau$, due to rollback.

Based on Eqs. (3) and (4), the following conclusions for the effectiveness of this operational model with regard to *VSC* elimination can be drawn:

- On receipt of ***n*** straggler- or anti-messages during a simulation cycle, the simulation process will be recovered after ***n*** subsequent simulation cycles;

• Additional processor time is spent on event processing, after which rollback actions are necessary;

• The messages about scheduled events, generated after event processing by SO, will be delivered by the parallel running CM before the beginning of the next simulation cycle. During the *VSC* elimination actions, these messages will be generated and sent many times due to the repeated event processing;

• Cancellation of the previous actions will lead to multiple increase of the number of outgoing anti-messages, as well as to increase of the probability for cascade rollback.

As a result, there is an increased synchronization overhead, which will result in degradation of the simulation process performance.

III. THE COST REDUCTION MODELS (CRM)

In the present Chapter, an operational model, aimed at decreasing of *VSC* elimination overhead, is proposed. The basic concept of this model is reduction of the number of necessary recovery simulation cycles. This can be achieved by merging of the actions for synchronization of different violator-messages into one single synchronization cycle.

The model consists of three components: communication, synchronization and simulation modules, implemented as threads running in parallel within a simulation cluster. The purpose of the communication module is the same as that presented in Chapter I. A substantial difference is the separation of the synchronization operations in a standalone synchronization module (SYN), aimed at achievement of parallel execution of event synchronization and event processing.

Let's consider the i[th] simulation cycle P_i, in which the event e_i is being processed (Figure 3).

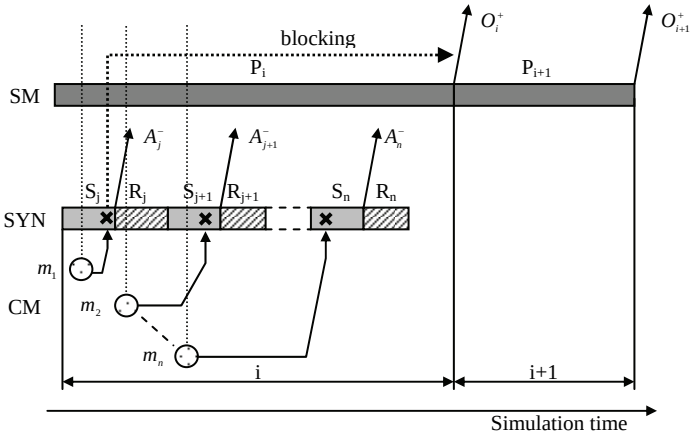

Fig.3. Multiple simulation correctness violation for CRM

Let m_τ be a violator-message $(\tau < CVT)$. Thanks to the parallel execution, SYN starts immediately a synchronization phase S_j. Meanwhile, the delivery of messages from SM to CM is blocked, which prevents sending of the messages through the communication network at the end of the processing phase. During S_j SYN starts rollback actions (R_j), resulting in recovery of the correct SO state.

Thus, in the i[st] simulation cycle, the number of sent anti-messages will be:

$$A_i^- = O^-(k_{CVT,\tau}), \quad (5)$$

where $O^-(k_{CVT,\tau})$ - number of anti-messages, necessary for canceling of the sent normal messages for all ***k*** processed events within the interval between the current *CVT* and the time τ of the violator-message.

Within the current cycle, due to the blockage, no normal messages will be sent, i.e. $O_i^+ = 0$.

VSC elimination will be executed during the next i+1[st] simulation cycle, wherein the simulation process will continue with processing of the *CEL*, already recovered to its correct state. During this cycle, no anti-messages will be sent ($A_{i+1}^- = 0$), while the number of outgoing normal event messages will be:

$$O_{i+1}^+ = O_i^+ + O^+(k_{CVT,\tau}) + O^+(\tau), \quad (6)$$

where O_i^+ - number of messages generated due to processing of event e_i from the i[th] simulation cycle;

$O^+(k_{CVT,\tau})$ - number of messages generated by the repeated event processing within the interval (CVT, τ), due to rollback;

$O^+(\tau)$ - number of messages generated due to processing of message m_τ. (In case of m_τ is an anti-message, $O^+(\tau)$ =0).

Similarly, for ***n*** violations, having occurred during the i[th] simulation cycle for a simulation time interval (τ, γ), where $\tau, \gamma < CVT$, the number of anti- and normal messages sent out can be represented:

$$A_n^-(i) = O^-(k_{CVT,\tau}) + O^-(k_{\tau,\gamma}), \quad (7)$$

where $O^-(k_{CVT,\tau})$ - number of anti-messages, necessary for canceling of the normal messages sent out for all processed ***k*** events within the interval between the current *CVT* and the time ***τ*** of the violator-message.

$O^-(k_{\tau,\gamma})$ - number of anti-messages, necessary for canceling of the normal messages sent out for all processed ***k*** events within the interval (τ, γ), in case of $\gamma < \tau$. In case of $\gamma \geq \tau$, this component has a value of zero,

and

$$Q_n^+(i) = O_i^+ + \sum_{e=1}^{n} O_e^+(t_{\tau \leq t \leq \gamma}) + O^+(k_{\tau,\gamma}), \quad (8)$$

where O_i^+ - number of generated messages by processing of event e_i from the i[th] simulation cycle;

$\sum_{e=1}^{n} O_e^+(t_{\tau \leq t \leq \gamma})$ - number of generated messages due to the processing of violator-messages within the interval (τ, γ). In case of anti-message, the value of this expression is zero;

$O^+(k_{\tau,\gamma})$ - number of messages generated by the repeated event processing within the interval (τ, γ), in case of $\gamma < \tau$, due to rollback.

Based on comparison of Eqs. (3)-(7) and (4)-(8), some conclusions about the positive results from application of this operational model can be drawn, mainly in the following directions:

- The proposed model allows *VSC* elimination for P_i to be done within one simulation cycle only, regardless of the number of violator-messages received during P_i;
- Reduction of the number of anti-messages, transmitted over the network. In case of a rollback, since the possibility for sending of normal messages at the end of the simulation cycle is blocked, the necessity of sending corresponding canceling anti-messages is also eliminated;
- Reduction of the number of outgoing repeated normal event messages. Thanks to the blocking mechanism in the rollback case, the initial sending of such event messages is ignored;
- Reduction of the total elapsed simulation time.

IV. Experimental evaluation and analysis

The experimental studies have been performed on the basis of the benchmark PHOLD [5] with increasing values of the number of SO and the events in the simulation model. PHOLD contains a fixed number N of simulation objects, connected through a 2D torus network and a set of tasks (messages) E, circulating among them. The increment of scheduled event's timestamps is determined by means of normal distribution. The destination of each newly generated scheduled event is chosen with equal probability among its four adjacent SO in the 2D topology.

The experiments have been carried out on a 100 Mb local area network of 900 MHz Pentium III PC workstations running Linux 2.6 operating system. A run-time system has been developed [6], consisting of simulation kernels performing the functions of simulation clusters. Each simulation kernel is executed as a standalone Linux-process on a separate workstation.

Fig. 4 shows the results, obtained within the study of the number of generated anti-messages

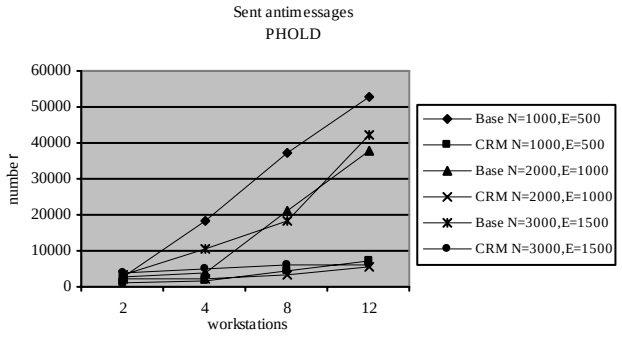

Fig.4. Sent anti-messages chart

A common tendency for both operational models studied (BASE and CRM) is the observed increase of the quantity of generated anti-messages proportional to the number of computational units. This is a normal consequence of the increased communication between the separate clusters wherein, because of message delivery delays, the potential probability of violator-message arrival, rollback and repeated message sending also increases. The experiments show clearly the better effectiveness of the CRM model with respect to the number of generated anti-messages, expressed as a considerable reduction of the latter. This is a direct consequence of the parallelism of the event synchronization and event processing actions, wherein, for the rollback case, thanks to blocking of process-generated normal event outgoing messages, the necessity of subsequent outgoing anti-messages for their cancellation is also eliminated.

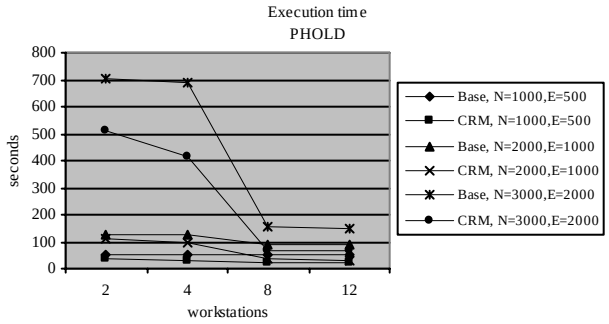

Fig.5. Execution time

Regarding the total elapsed simulation time, the CRM model demonstrates better features, as shown on Fig. 5. These better features are mainly due to the considerable reduction of the number of outgoing anti-messages, and the reduction of repeated event processing overhead, as well.

V. Conclusion

This paper presents a study of operational models for a Time Warp discrete-event simulation over a network of workstations. The experimental results show definitely the better features of the proposed CRM model regarding *VSC* elimination performance overheads. As a future goal for research work, improvement of the effectiveness of the proposed model with regard to reduction of performance overheads for recovery of previous correct simulation states should be pointed out first of all.

References

[1] Fujimoto R.M. Parallel Discrete Event Simulation. Communications of the ACM, vol.33, no.10, pp.41-52, 1990.

[2] Nance R., Sargent R. Perspectives on the Evolution of Simulation. Operations Research, *INFORMS*, vol.50, no.1, pp.161-172, 2002.

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

[4] Bagrodia, R., R. Meyer, Y. Chen. Parsec: a parallel simulation environment for complex systems, IEEE Computer, vol.10, pp.77-83, 1996.

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

[6] Valchanov H., Ruskova N., Ruskov T. Implementation of a Language for Distributed Simulation, Proc of the Computer Science Conference'05, Chalkidiki, Greece, vol.I, pp.208-213, 2005.