REGENERATIVE QUEUING NETWORK DISTRIBUTED SIMULATION

Panajotis Katsaros and Constantine Lazos
Department of Informatics
Aristotle University of Thessaloniki
54006 Thessaloniki
Greece
E-mail: {katsaros, clazos}@csd.auth.gr

KEYWORDS
Queuing models, Performance analysis, Statistical analysis, Distributed simulation.

ABSTRACT

Complex, closed or open queuing network simulations, has been proved to be particularly time expensive experiments, when an acceptable level of estimation accuracy is to be achieved. Moreover, in most cases, the same experiment has to be repeated many times, by varying the model's parameters. Thus, a distributed execution architecture is often adopted as more suitable. At least two different types of distributed discrete event simulations have been suggested. Their emphasis lies on the development of the appropriate synchronization, deadlock handling and/or memory management mechanisms. Our approach, moves the focus of interest, to the exploitation of the statistical nature of the simulation experiment, to obtain more reliable results. Thus, the regenerative property and its presence in queuing network simulations is studied. We suggest a way of distributing computation in regenerative queuing network simulation and we prove that, our approach, is theoretically correct. The method is applicable in a broad class of queuing network simulations with significant gains.

1. INTRODUCTION

The field of parallel and distributed simulation has grown rapidly over the last years as a result of the need for simulating complex models with significant execution times. In particular, parallel/distributed discrete event simulation has been widely studied and at least two different types of execution mechanisms have been successfully implemented. Simulation models where these execution architectures can be easily applied include:

- Petri net models (Petri 1962; Peterson 1981), which are usually used to represent synchronization between various concurrent activities,
- Cellular automation models (Toffoli and Margolus 1987)
- Queuing networks, which are usually used to represent contention for the multiple resources that comprise a system.

In an event based queuing network simulation, the logic associated with each event type is incorporated into the simulator (e.g. job departure, arrival events e.t.c.). A simulation run then takes place by executing each scheduled event in a time-ordered sequence. For this reason, a single data structure called event list, is usually used to hold future events. The event list is sorted by event time and the simulation execution involves removing events from the head of the list and causing the action associated with each event to be performed. When an action causes scheduling of new events, these events are inserted into the event list.

In parallel/distributed simulation, one of the main characteristics is the partitioning of the simulation model into a number of sub models. These sub models are called *logical processes* (LP). Thus, in the queuing network distributed simulation, each LP is defined as a set of one or more queues and a Future Event List (FEL).

In the *conservative asynchronous distributed simulation* (Chandy and Misra 1979; Lin and Fishwick 1995), which is also known as the *Chandy-Misra distributed discrete event simulation*, each LP, runs as a separate task on one of the processors that are used. Its FEL is just used for the so-called internal events, i.e. events scheduled for execution within the same LP. An attribute called Local Virtual Time (LVT) represents the time-stamp of the event that just occurred in the LP. Execution of external events should not violate the local causality constraint which prescribes that events are processed in no decreasing time-stamp order.

Thus, in a Chandy-Misra queuing network simulation each LP performs event processing based on events in the FEL, but also processes external events from the corresponding input channels, if the minimum next external event time, is less than the next internal event time (if any). Because of the local causality constraint, an LP does not process any input message until it has received at least one message from each of its input channels. Thus, this execution mechanism has the potential for deadlock. This happens, when an LP blocks processing because there are not external events available on all input channels. The LP does not produce any output events and this results in the successive blocking of the other LPs. Two deadlock resolutions have been proposed: deadlock avoidance (Chandy and Misra 1979) and deadlock detection/recovery (Chandy and Misra 1981; Misra 1986).

The most recently suggested variant of asynchronous distributed discrete event simulation, is known as the *optimistic* approach (Jefferson 1985; Jefferson and Sowizral 1985) or *the Time Warp mechanism*. Optimistic simulation strategies, in contrast to conservative ones, do not strictly adhere to the local causality constraint. Instead of this, LPs allow the occurrence of causality errors and provide a mechanism to recover from them. More precisely, if an external event arrives and the timestamp of the message is less than some of the events already executed (*straggler* *message*), then the LP rolls back to the most recently saved state in the simulation history consistent with the timestamp of the arriving external event. Then, simulation restarts from that state on, as a matter of local causality constraint violation correction. If in the meanwhile the LP has already sent one or more output messages since the time instant where the LP rolls back, then an *antimessage* is sent for each of them.

Each LP has to keep sufficient information, say past state buffers, past input buffers, antimessage buffers etc., in order to be able to roll back. However, this means, that a possible memory limitation may cause the protocol to execute fairly slow. For this reason, a memory management algorithm is usually applied in order to guarantee availability of a "sufficient" amount of memory.

In this work it will be shown, how a preliminary study for the existence of a statistical property called regeneration, can lead to a simpler execution mechanism, which also produces more accurate results. Our approach basically moves the focus of interest from the development of synchronization, deadlock handling and/or memory management mechanisms, to the exploitation of the statistical nature of the simulation experiment.

2.QUEUEING NETWORK SIMULATION AND THE REGENERATIVE PROPERTY

In most cases of queuing network simulations, the main concern is the estimation of one or more performance measures under the assumption that the system behaves as it is in an equilibrium condition. The reason is, that it is usually desirable to obtain reliable estimations, by discarding the bias, introduced by the arbitrary selection of the initial state of the system. There are two different alternatives:

- To calculate the extent of the transient phase of a simulation run and then to discard the measurements obtained in this time period.
- To justify that the system is initialized in a state, which can be considered to represent it, in a particular time instant within its equilibrium (steady-state) phase.

However, even in the case, where the problem of the initial transient phase has been overcomed, the same simulation will give slightly different results in a second run. This happens, because every simulation is essentially a random experiment. For this reason, analysts are usually interested in obtaining mean values and confidence intervals of the required performance estimates. If our aim is to utilize the tools provided by the theory of statistics, the problem takes the form of producing a number of observations independent to each other. The following approaches are available for this purpose:

- To carry out a number of *independent replications* of the simulation experiment. Then, estimation of the extent of the transient phase of each experiment has to take place, in order not to take into account the observations of this time period to the calculation of the mean values.
- To find a recurrent state (if any), which can be considered to represent the system in a particular time instant within its equilibrium phase. Every time the simulation passes through such a state, the experiment is probalistically restarted. Thus, a single simulation run is partitioned into cycles. At the end of each cycle, a new observation of each one of the required performance measures can be estimated and listed. All these observations are independent to each other and they can be used, in producing confidence intervals. This method is known as the *regenerative* method.
- *Batch means* and *spectral method* (Lavenberg 1983) are less in use and in any case are beyond the scope of this work.

The first approach, requires a time consuming process to take place, in order to statistically analyzing the obtained observations. Moreover, a vast amount of data has to be thrown away. If the second approach is to be applied, a sound theoretical basis for the existence (or not) of the regenerative property in the simulation model has to be used.

This basis is provided in (Shedler 1993) and it can be summarized as follows. Let us assume that, at any time instant, each job, is of exactly one class and one type. Jobs may change class as they traverse the network, but they can not change type. The type of a job may influence its routing path through the network as well as its service requirements at each service center. Service priorities can also be associated with job types. Every discrete event simulation on a finite or countably infinite state space is a *generalized semi-Markov process* (GSMP)* and so, the regenerative method is applicable, if this process is characterized by the regenerative property. Assuming a state s, where all jobs, are placed at a service center, which sees only one class, or is such that, jobs of the lowest priority are subject to pre-emption, let us call D, the set of all states, accessible from s. The GSMP which is restricted to the set D, has been proved, to be characterized by the regenerative property. Let us assume, our aim is to estimate the mean value of a queuing network characteristic (e.g. throughput), which is given, as a real-valued function f over the regenerative stochastic process $X = \{X(t)\ ;\ t \geq 0\}$

$$k(f) = E[f(X)]$$

Let us also call

$$Z_k(f) = \int_{T_{k-1}}^{T_k} f(X(u)) \cdot du$$

the observation produced by the kth regenerative cycle.

* The GSMP is defined as a stochastic process that makes a state transition when an event associated with the occupied state occurs. Several possible events, associated with a state, compete with each other to trigger the next transition. At each transition of the GSMP, new events may be scheduled. For each of these new events, a clock indicating the time when the event is scheduled to occur, is set, according to an independent (stochastic) mechanism. If a scheduled event does not trigger a transition but is associated with the next state, its clock continues to run; if such an event is not associated with the next state, it ceases to be scheduled and its clock reading is abandoned.

Shedler proved, that a 100 α % confidence interval for k(f), after the completion of N regenerative cycles, is given by

$$\left[\hat{k}(N) - \frac{s(N)\cdot F^{-1}\left(\frac{1+a}{2}\right)}{\sqrt{N}\cdot\bar{\tau}(N)}, \hat{k}(N) + \frac{s(N)\cdot F^{-1}\left(\frac{1+a}{2}\right)}{\sqrt{N}\cdot\bar{\tau}(N)}\right]$$

where

$\bar{\tau}(N)$ is the average cycle length

and

$$\hat{k}(N) = \frac{\bar{Z}(N)}{\bar{\tau}(N)}$$

$$s^2(N) = s_{11}^2(N) - 2\hat{k}(N)s_{12}^2(N) + (\hat{k}(N))^2 s_{22}^2(N)$$

with

$$s_{11}^2(N) = \frac{1}{N-1}\sum_{k=1}^{N}(Z_k(f) - \bar{Z}(N))^2,$$

$$s_{22}^2(N) = \frac{1}{N-1}\sum_{k=1}^{N}(\tau_k - \bar{\tau}(N))^2$$

$$s_{12}^2(N) = \frac{1}{N-1}\sum_{k=1}^{N}(Z_k(f) - \bar{Z}(N))(\tau_\kappa - \bar{\tau}(N))$$

In (Katsaros and Lazos 2000), we have described a technique for determining the length of a queuing network regenerative simulation. The algorithm was successfully used in the queuing network simulator, that we have developed for carrying out simulation experiments of complex closed and open queuing network models. More precisely, if the desired accuracy will be determined by δ, so that the half length of the 100 α% confidence interval to be obtained will be not more than 100 δ% of k(f), then the number N of the required regenerative cycles can be dynamically determined as follows

$$N \geq \left(\frac{F^{-1}\left(\frac{1+a}{2}\right)}{\delta}\right)^2 \cdot \left(\frac{s(l)}{\hat{k}(l)\cdot\bar{\tau}(l)}\right)^2 \qquad \text{(A)}$$

where $s(l), \hat{k}(l), \bar{\tau}(l)$ are the sample estimates, after the l th cycle of the simulation experiment. The simulation terminates, when the maximum number N of regenerative cycles (for all the queuing network estimators and for every queue in the network) has been completed.

Figure 1, shows a closed queuing network model given in (Sauer and Chandy 1981), which has been simulated by successfully applying the regenerative method. The model represents an interactive computer system. There are ten users at terminals. Each user thinks for a moment, keys in a command and waits for response. Upon receiving a response, the user repeats this cycle. Let us assume that the mean time for thinking and keying, has an exponential distribution with mean 3 seconds. The processor sharing discipline, has been used for the terminals, since there is always a server for each job. The passive queue which has been included in the model, represents memory contention. Memory is divided into 4 partitions and each token represents a partition. A 50% of the jobs arriving at the CPU are able to produce another job, which can do I/O, while the creating job is still at the CPU. After either of these overlapped activities is finished, the process which is finished, is forced to wait for the other to finish. The regenerative state was defined as the state, where all jobs are placed at the terminals. Finally, the maximum half length of the 90% confidence intervals to be produced, was required to be no more than 4% of the estimated value.

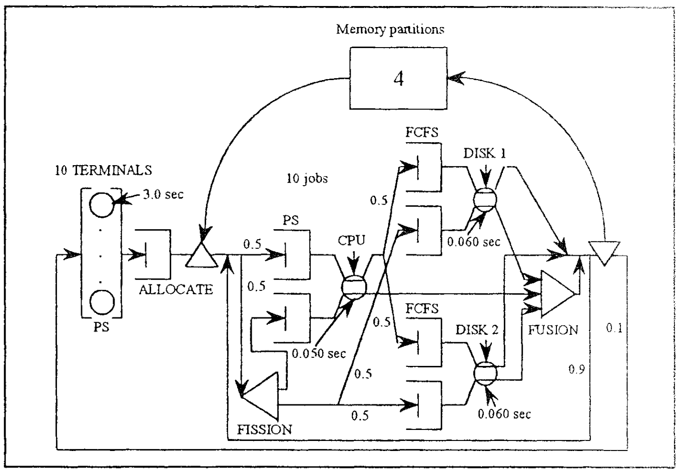

Figure 1 An interactive computer system with memory contention and process overlapping

The results obtained are as follows

```
SIMULATED TIME:                    2938 sec
NUMBER OF EVENTS:                  106853
NUMBER OF CYCLES:                  132
AVERAGE NUMBER OF EVENTS:          809.492
AVERAGE CYCLE LENGTH:              22.087 sec
90% CONFIDENCE INTERVAL:           (17.311, 26.862)

                    RESULTS FOR THE SERVICE ACTIVITIES
                  THROUGHPUT jobs/sec    UTILIZATION
                   LB     MEAN     UB     LB     MEAN    UB
TERMINALS         1.688  1.715  1.743 5.174  5.214  5.255
ALLOCATE          1.688  1.715  1.743 3.468  3.534  3.601
CPU              16.955 17.330 17.705  .848   .867   .886
DISK1             8.563  8.741  8.920  .509   .518   .527
DISK2             8.409  8.589  8.768  .503   .513   .523
                 RESPONSE TIME sec
                  LB     MEAN    UB
TERMINALS        2.977  3.040  3.102
ALLOCATE         2.681  2.790  2.899
CPU              .129   .131   .133
DISK1            .098   .100   .103
DISK2            .098   .099   .101

                   RESULTS FOR THE QUEUES
                  TOTAL LENGTH jobs
                  LB     MEAN    UB
TERMINALS        5.174  5.214  5.255
ALLOCATE         4.722  4.786  4.849
CPU              2.241  2.275  2.309
DISK1             .866   .876   .887
DISK2             .840   .852   .864
```

Figure 1, illustrates the fact that application of the regenerative method does not necessarily require compromises to the structure complexity of the simulated model. If however, a large number of jobs will be placed in the simulated model, this will result in an increase of the average number of events per cycle and so, in longer regenerative cycles. Moreover, if our aim is to achieve better confidence interval lengths, the number of regenerative cycles that have to be completed, increase respectively.

In open queuing networks, where a steady-state distribution exist, it is necessary to use observations of a large number of regenerative cycles, if confidence intervals of an acceptable length is to be produced. This is justified by the variability in the total number of jobs, which results in a highly variable system.

3. DISTRIBUTED REGENERATIVE QUEUEING NETWORK SIMULATION

From what has been presented so far and from the simulation experiments conducted, it is shown, that

- Simulated time is not important, from the point of view that the experiment depends only on the return of the model to the same state, irrespective of the time instant that this will happen. Certainly, the statistics collected at the end of each regenerative cycle, are dependent on the difference between the time instant of the end and the time instant of the start of the cycle. However, this is basically a random process that is probabilistically restarted at the end of each regenerative cycle.
- The method can be applied irrespective of the structure complexity of the simulated model, under the assumption of validity of a set of conditions that have been described in §2.
- The method, can lead in particularly time-consuming experiments, when an acceptable level of estimation accuracy is to be achieved. The problem becomes even worse, in cases where the same experiment has to be repeated with varied parameters for sensitivity study purposes.
- The stopping algorithm which is based on (A) and has been used in our simulator, is a suitable means for controlling the execution of a regenerative simulation.

All these, prove the fact, that an execution architecture, that combines independent simultaneous replications and the regenerative method, is not only desirable, but it is also a feasible alternative. The results obtained by the use of the regenerative method are only based on the statistics collected at the end of each regenerative cycle. Thus, the main concern, is to complete as many regenerative cycles as need, in order to obtain the required confidence interval lengths.

In the distributed regenerative queuing network simulation, each LP contains the entire queuing network model. A number of LPs start the execution of the experiment simultaneously. The simulation stops when the sum of the regenerative cycles completed by each of the LPs, satisfies (A) for all the queuing network estimators and for every queue in the network.

4. FUTURE WORK

Distributed regenerative queuing network simulation, is not burden with extra communications load, because of the application of a synchronisation scheme. Termination however, has to be controlled by an algorithm which will be based on (A) and can work in the frame of a central coordinator process, or by the use of a circulating values-carrying token.

In the first case, each LP, sends the statistics collected (after the end of the last regenerative cycle) to a central coordinator process. The coordinator process checks if (A) is satisfied for all the queuing network estimators and for every queue in the network. If so, a message is sent to all LPs. On reception of this message, each LP stops and the results produced since the last regenerative cycle are discarded. Finally, the coordinator process produces a simulation report.

In the second case, we distinguish two different implementation possibilities:

- The token, circulates, through the alive LPs, constantly by collecting statistics for the already completed regenerative cycles. As soon as the token detects the end of the experiment, changes its color to black and continues its journey for one more time. On reception of the black token, each LP terminates and simulation results produced since the last visit of the token are discarded. The simulation report is then generated by the LP where the token stopped. This approach is similar to the Misra's algorithm (Misra 1983) for detecting termination. It is important that this algorithm makes no assumptions whatever, about the topology of the communication channels, nor about the transit times of the messages. Its only assumptions are, that, no messages are lost and that messages are always received in the order in which they are sent. All the LPs behave identically and there is no process with a special role. Any LP can start a detection procedure by launching a token, and can label this, with its own identity, as to avoid confusion with any tokens issued by other processes. However, this may happen, only after the completion of the first regenerative cycle and under the condition that the LP has not already accepted the visit of another token labeled with the identity of a different LP. If, however, there is already one or more tokens which circulate through the LPs and they have not yet arrived to that specific LP, they will be simply ignored when they arrive. Thus, in the end, there will be only one token circulating through the LPs for detecting termination of the experiment.
- The token changes position at the time instants that a regenerative cycle is completed in any LP. The LP broadcasts this event and waits for the reception of the values-carrying token. This implementation approach, assumes, that broadcasting is supported by the underlied communication system.

Finally it is worth to note, that the distributed regenerative architecture retains the possibility of implementing algorithms, for obtaining derivatives of expectations with respect to various parameters (Reiman and Weiss 1986). This allows performing model sensitivity studies and optimization based on the results of a single simulation run.

5. RELATED WORK

Unfortunately, exploitation of statistics for carrying out parallel/distributed discrete event simulation experiments, has so far, been overlooked. Moreover (Pawlikowski 1990), in most cases, the results of simulation studies have little credibility, since they are presented without regard to their random nature and the need for proper statistical analysis of simulation results.

The only work known to us which is related to our approach is the one by Raatikainen (Raatikainen 1992).

6. CONCLUDING REMARKS

In this paper, we present the most important architectures for performing parallel/distributed discrete event simulation experiments. The conservative and the optimistic approach, constitute two different alternatives for distributing computations in queuing network simulations.

However, since every discrete event simulation is basically a random experiment, the estimation of the required characteristic(s) of the simulated model is therefore a statistical process. Thus, we present the underlied theoretical framework of the regenerative method for analyzing the simulation output data.

Having noticed, that a regenerative simulation experiment, depends only on the return of the model to the same state (irrespective of the time instant that this will happen), we conclude, that an execution architecture that combines independent simultaneous replications and the regenerative method is feasible, if an appropriate control scheme can be applied. Experimental results have shown that, (A) provides a suitable means for determining a simulation run length and can be implemented in an either centralized or distributed way.

Moreover, the new architecture is free of added communication load, that is caused by the application of a process synchronisation scheme.

Our approach, is innovative, in that, it exploits a statistical property which may be present in queuing network simulation models and this helps to distribute computations in a more efficient way. To conclude, a thorough study of the statistical nature of every simulation model should take place, and this will lead in more efficient and accurate experiments.

REFERENCES

Chandy, K.M. and Misra, J. 1979. "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs" *IEEE Trans. on Software Engineering*, Vol. SE-5, no. 5 (Sept.): 440-452.

Chandy, K.M. and Misra, J. 1981. "Asynchronous Distributed Simulation via a Sequence of Parallel Computations" *Communications of the ACM* 24, no. 11 (Apr.): 198-206.

Jefferson, D.A. 1985. "Virtual Time" *ACM Transactions on Programming Languages and Systems* 7, no. 3 (July): 404-425.

Jefferson, D. and Sowizral, H. 1985. "Fast Concurrent Simulation Using the Time Warp Mechanism" In: *Reynolds P., Ed., Distributed Simulation* SCS - The Society for Computer Simulation, Simulation Councils, Inc., La Jolla, California, 63-69.

Katsaros, P. and Lazos, C. 2000. "A technique for determining queuing network simulation length based on desired accuracy" *Journal of Computer Systems Science and Engineering*, CRL Publishing, (to appear).

Lavenberg, S.S. 1983. *Computer Performance Modeling Handbook.* New York, Academic Press.

Lin, Y.-B. and Fishwick, P.A. 1995. "Asynchronous Parallel Discrete Event Simulation" *IEEE Trans. on Systems, Man and Cybernetics*, Vol. XX, No. Y.

Misra, J. 1983. "Detecting Termination of Distributed Computation Using Markers" In *Proc. of the 2^{nd} annual ACM Symposium on Principles of DC* (Montreal, August 1983), 290-294.

Misra, J. 1986. "Distributed Discrete-Event Simulation" *Computing Surveys* 18, no. 1 (March): 39-65.

Pawlikowski, K. 1990. "Steady-State Simulation of Queuing Processes:A Survey of Problems and Solutions" *ACM Computing Surveys* 22, No. 2 (June): 123-170.

Peterson, J.L. 1981. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Inc., Englewood Cliffs, N.J.

Petri, C.A. 1962. *Kommunikation mit Automaten.* Bonn: Institut fur Instrumentelle Mathematik. Schreften des IIM No. 2.
Also in English translation:
Petri, C.A. 1966. "Communication with Automata" Technical Report RADC-TR-65-377, Vol 1, Suppl 1. Applied Data Research. Princeton, NJ. Contract AF 30(602)-3324.

Raatikainen, K. 1992. "Run Length Control using Parallel Spectral Methods" In *Proceedings of the 1992 Winter Simulation Conference* (December 1992).

Reiman, M. and Weiss, A. 1986. "Sensitivity Analysis Via Likelihood Ratios" In *Proceedings of the 1986 Winter Simulation Conference.*

Sauer, C.H. and Chandy, K.M. 1981. *Computer Systems Performance Modeling.* New Jersey, Prentice-Hall.

Shedler, G. 1983. *Regenerative Stochastic Simulation.* California, Academic Press.

Toffoli, T. and Margolus, N. 1987. *Cellular Automata Machines: A New Environment for Modeling.* 2^{nd} edition. MIT Press.

BIOGRAPHY

Panajotis Katsaros received a BSc degree in mathematics from the Aristotle University of Thessaloniki – Greece, in 1992. He has also received an MSc degree in software engineering from the University of Aston, Birmingham – UK, in 1993. He is currently a PhD candidate in the Department of Informatics of the Aristotle University of Thessaloniki – Greece and he is supervised by Professor Lazos. His research interests include queuing network performance analysis and simulation of distributed object systems and advanced sensitivity analysis techniques.