

ANALYSIS OF OUTPUT DATA IN DISTRIBUTED SIMULATION

B. Vinod Kumar Reddy
Jayendran Venkateswaran

Industrial Engineering and Operations Research
Indian Institute of Technology Bombay
Powai, Mumbai 400076, INDIA

ABSTRACT

This paper presents some preliminary results from the ongoing work on output analysis in distributed simulation. A scheme to analyze system performance using the distributed simulation models' output is discussed. Also, the importance on the choice of random number streams is illustrated.

1 INTRODUCTION

In the past few decades there has been growing interest in the area of distributed simulation to analyze system of systems. Since the early 1980s efforts had been directed towards enabling distributed simulation by tackling issues of data and time management, culminating in the development of IEEE Standard 1516 High Level Architecture (HLA) in 2000. Since 2000, the focus of works include enabling infrastructure for manufacturing simulations (McLean and Riddick 2000;), interface specifications for Commercial-Off-The-Shelf (COTS) simulation packages (Taylor 2003, Garg et al. 2009), and case studies (Mustafee et al. 2006, Cao et al. 2007). Mustafee et al. (2006) have compared the performance of conventional (single) simulation and distributed simulation with respect to execution time as the number of modules increases.

The main purpose of any simulation (single or parallel or distributed) of stochastic systems is to help estimate the 'true' characteristics of the system. Schemes for proper statistically analysis of single (non-distributed) simulation output process has been well established (Law and Kelton 2000). One key aspect which has not been discussed enough in academic circles is the issues related with the analysis of distributed simulation output data. Kremien et al. (1989) had presented some statistically guidelines for analyzing output data of distributed system implementing load-sharing algorithms. However, their work is more applicable to a parallel simulation rather than geographically distributed simulation systems.

2 OUTPUT DATA ANALYSIS

Classical statistical techniques cannot be directly applied to the output of a single replication of a simulation model since the output processes of all simulations (even non-distributed) are non-stationary and autocorrelated. Hence, the concept of *independence across runs* is used to obtain independent and identically distributed (IID) observations. That is, in each run (or replications) different random numbers are used, resulting in different IID realizations of the output random process. Let X_j be the sample random variable observed on the j th replication. Suppose n replications are made. The $100(1-\alpha)\%$ confidence interval (CI) estimate of the true mean $\mu = E(X)$ can be given as:

$$\bar{X} \pm t_{n-1, 1-\alpha/2} S / \sqrt{n}$$

where \bar{X} is the sample mean and S is the sample standard deviation.

Broadly the measures of interest are classified as tally statistics, time persistent statistics or counter statistics (Law and Kelton 2000). Tally or observation based statistics are obtained using a list of discrete observations. For example average time spent in system, maximum time in queue. Time-persistent or duration statistics is obtained by taking time average value of the variable. For example average number in queue, machine utilization. Counter statistics are accumulated sums of the variable. For example, number of parts produced.

2.1 Analyzing Output in Distributed Simulation

Now, suppose a system (say, manufacturing shop floor) is modeled as k distributed simulation sub-models that interact with each other dynamically. If the measure of interest wholly resides within a single sub-model (say, utilization of an unshared resource) then the exact same methodology for statistical analysis can be employed as done when analyzing data from a single simulation model. Taylor (2003) and Garg et al. (2009) have specified several modes in

which distributed simulation models interface with each other. Broadly, these interfacing can be to transfer entities, share resources, share buffer space / conveyors, share data and events between the different models. Due to the presence of such interfaces between the multiple distributed simulation models, additional care is required in analyzing distributed simulation output data.

Consider a case where one is interested in estimating the average time spent in system of entities flowing across multiple models. One way to estimate the time spent is to record each entity's arrival time into the system in Model_1 and the time it leaves the system in Model_k. This method will provide an estimate of the average time in system only if all the entities have the exact same pre-determined flow. However, entities can enter and/or leave the system from more than one model. In such cases, a central coordinator may be required for statistical data analysis, which defeats the purpose of distributed simulation.

A straightforward method will be to separately measure the statistics for each of the distributed models, and then sum the measures to obtain the system-wide performance. For instance, let a system be distributed into two simulation models: Model A and Model B. Let X_j^a and X_j^b be (say) the average time in system measure on the j th replication from Model A and B respectively. Now, the average time in system X_j can be simply obtained by summing the variables, $X_j = X_j^a + X_j^b$. Further,

$$X_j^a = \sum_{i=1}^{ma_j} X_{ij}^a / ma_j \text{ and } X_j^b = \sum_{i=1}^{mb_j} X_{ij}^b / mb_j,$$

where ma_j and mb_j are the number of observations in j th replication in Model A and Model B, respectively. Similarly, let U_j^a and U_j^b be (say) the measure of utilization of a shared resource on the j th replication from Model A and B respectively. Now, the total utilization U_j of the shared resource can be given as, $U_j = U_j^a + U_j^b$, with

$$U_j^a = \int_0^{Ta_j} U_j^a(t) / Ta_j \text{ and } U_j^b = \int_0^{Tb_j} U_j^b(t) / Tb_j,$$

where Ta_j and Tb_j are the simulation duration in the j th replication in Model A and Model B, respectively.

This above proposed scheme, though straightforward, raises two issues:

- The number of observations in case of tally statistics, or the duration in case of time persistent statistics maybe different for the different models. Statistically, this should not have significant impact on the system measure.
- The choice of random numbers used in Model A and Model B could induce correlation on the system performance measure. That is, $\text{Var}(X_j) = \text{Var}(X_j^a + X_j^b) = \text{Var}(X_j^a) + \text{Var}(X_j^b) + 2\text{Cov}(X_j^a, X_j^b)$. The issue of the choice of random number on

the results of the distributed simulation is further investigated in this paper.

3 SCHEMES FOR USE OF RANDOM NUMBERS

Generation of IID Unifom(0, 1) random numbers and its use in generating random variates form the basic step in capturing randomness in stochastic simulations. In distributed simulation, the following three types of use of random numbers arise:

- *Use of default random number streams:* This refers to the use of the 'simulation package default' random number streams in each model. That is, if two simulation models A and B are built using the same package, then by default, both models will use the exact same stream of random numbers albeit for different purposes. Intuitively, the use of such common random numbers can induce correlation in the simulation models' outputs.
- *Use of dedicated random number streams:* This refers to the use of distinct non-overlapping stream of random numbers for capturing each source of randomness across all models. That is, if Model A has 7 sources of randomness (say arrival times) then random number streams 1 to 7 are assigned to the 7 arrival times. If Model B has 3 other sources of randomness (say processing times), then random number streams 8 to 10 are assigned to the 3 process times. This may be difficult to implement since the number of stochastic variables in the system may be quite large, and requires a central coordination mechanism which may not be possible in distributed setting
- *Use of separate random number streams:* This refers to the use of separate 'simulation package default' random number streams for each model such that each model uses non-overlapping random streams. This reduces wastage of random numbers.

Each of the above schemes is tested for a simplistic distributed queuing scenario, as discussed in the following section. It is noted that the above schemes should have no significant effect in a single (non-distributed) model, and is applicable to distributed models only.

4 SAMPLE QUEUEING SCENARIO

A production line with 3 $M/M/1$ queues in series is considered, as shown in Figure 1. Each server is of unit capacity and the queues are of infinite capacity. Entities arrive at the first queue at an average rate $\lambda=3/\text{hr}$, and the serviced at the rates of $\mu_1=5/\text{hr}$, $\mu_2=10/\text{hr}$ and $\mu_3=60/\text{hr}$ at the three servers respectively. The departure process of an $M/M/1$ type

queue is also Poisson with the same rate as the arrival process, as $t \rightarrow \infty$. The performance of the tandem queuing system can be computed analytically. The steady state utilizations of the 3 servers are found to be 0.6, 0.3 and 0.05 respectively. And the steady state average time spent in system is 39.83 minutes.

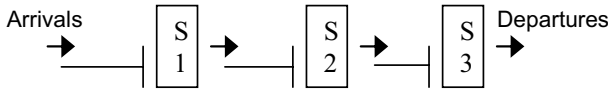


Figure 1: Sample queuing scenario

5 DISTRIBUTED SIMULATION

The queuing scenario is implemented in distributed simulation, with the arrival process and server 1 in Model A, and server 2, server 3 and departures in Model B; as shown in Figure 2. Entity transfer takes place from Model A to Model B. The interface specification between Model A and Model B used is asynchronous entity transfer (Garg *et al.* 2009). Sequence diagram for asynchronous entity transfer is shown in Figure 3, where Model A considered as Sender model and Model B as Receiver model.

When Model A is ready to transfer an entity, the Model A sends an ‘Entity Transfer’ message to Model B. The ‘Entity Transfer’ message contains all the required information about the entity and its attributes. Model B decodes the message and creates the entity with mentioned attributes. At same instant, the entity is deleted from the Sender model.

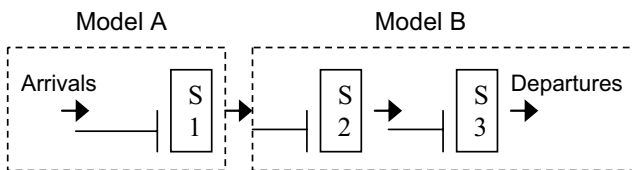


Figure 2: Distributed models

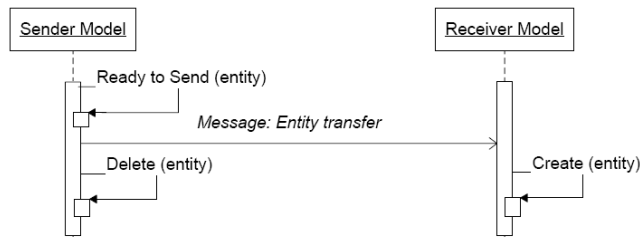


Figure 3: Sequence diagram for asynchronous entity transfer

5.1 Enabling Distributed Simulation

A client server based infrastructure has been used for the integration of distributed simulation models. The components of the infrastructure are: (1) Simulation models or federates, (2) Interface DLL, and (3) Transaction Coordinator. The simulation federates communicate with each other using XML-based messages via the Transaction Coordinator. Federates, distributed across the internet, interface with the Transaction Coordinator using an Interface DLL. The main function of the Transaction Coordinator (implemented in Java) is to facilitate time and message management between the distributed models. A conservative event based time synchronization scheme has been implemented.

6 RESULTS AND ANALYSIS

The statistics as obtained from a single simulation model is compared with that obtained from the distributed simulation models. For the given example, there are four sources of randomness in the system: inter-arrival times – Expo(20), Server 1 service time – Expo(12), Server 2 service time – Expo(6), Server 3 service time – Expo(1). All times are in minutes. In distributed simulation there will be two sources of randomness each in Model A and Model B. The key statistic of interest is the average time spent in system. In case of single simulation, the average time spent in system is determined directly from simulation. In distributed simulation, the outputs ‘time spent in Model A’ and ‘time spent in Model B’ are obtained, using which the time in system is computed as explained in Section 2.1. The results have been obtained over 5 replications, with each replication of 20000 minutes length. All simulation models are built using the COTS package ArenaTM.

Output from single simulation and distributed simulation are discussed for the three different cases of random numbers streams used for generating input values.

6.1 Output from use of Default Streams

The simulation package is allowed to use the default stream of random numbers in both single simulation as well as distributed simulation. A stream refers to a particular sequence of random numbers. In the case of single simulation model, the default stream is used for all 4 input distributions. In the case of distributed simulation, Model A and Model B uses the default stream for their 2 inputs respectively.

The 95% CI for time in system statistic for single and distributed models in case of default streams is as shown in Table 1. The average time spent in system computed analytically (see Section 4) is 39.83 minutes. This steady state value falls within the 95% CI as obtained from single simulation and distributed simulation. Further, it is ob-

served that the expected average time in system obtained from distributed simulation is 41.07 is much larger than that obtained from single simulation (35.76).

Table 1: Average time in system results (default streams)

	Distributed Simulation			Single simulation: Time in system
	Time in Model A	Time in Model B	Time in system	
Mean	32.14	8.93	41.07	35.76
Std.Dev	3.56	0.21	3.57	3.34
95% CI	(25.73, 33.13)	(8.81, 10.75)	(36.64, 45.50)	(31.61, 39.91)

6.2 Output from use of Dedicated Streams

In the simulation models, each input distribution is assigned different non-overlapping random streams for use. $\text{Expo}(\lambda, \langle \text{stream} \rangle)$ represents an exponential distribution with mean λ using specified stream of random numbers for the purpose of generating variates. In the single and distributed simulations, the input distributions are specified as follows: inter-arrival times – $\text{Expo}(20, 7)$, Server 1 service time – $\text{Expo}(12, 8)$, Server 2 service time – $\text{Expo}(6, 9)$, Server 3 service time – $\text{Expo}(1, 10)$.

The 95% CI for time in system statistic for single and distributed models in case of dedicated streams is as shown in Table 2. Since dedicated streams are used, it is to be expected that the results from single simulation and distributed simulation should be exactly the same. However, there is a slight difference in the results which is attributed to the message communication lag. Also, the analytical steady state value (39.83) falls within the 95% CI as obtained from single simulation and distributed simulation, with the mean quite close to the analytical value.

Table 2: Average time in system results (dedicated streams)

	Distributed Simulation			Single simulation: Time in system
	Time in Model A	Time in Model B	Time in system	
Mean	29.95	9.69	39.65	39.43
Std.Dev	3.57	0.59	3.62	3.25
95% CI	(25.51, 34.39)	(8.95, 10.43)	(35.14, 44.15)	(35.38, 43.47)

It is also noted that the mean utilization of the servers obtained from single and distributed simulation were the same (server 1 – 0.65, server 2 – 0.30, server 3 – 0.05).

6.3 Output from use of Separate Streams

In single simulation models, same stream (stream 8) of random numbers is used for all the four input distributions. In case of distributed simulation, Model A samples from

stream 7 and Model B samples from stream 9 for all their respective input distributions.

The 95% CI for time in system statistic for single and distributed models in case of separate streams is as shown in Table 3. The analytical steady state value (39.83) falls within the 95% CI as obtained from single simulation and distributed simulation, with the mean quite close to the analytical value. The mean utilizations as obtained from distributed simulation are: server 1 – 0.60, server 2 – 0.29, server 3 – 0.05.

Table 3: Average time in system results (separate streams)

	Distributed Simulation			Single simulation: Time in system
	Time in Model A	Time in Model B	Time in system	
Mean	29.43	9.78	39.21	40.063
Std.Dev	2.98	0.78	3.08	3.76
95% CI	(25.73, 33.13)	(8.80, 10.75)	(35.39, 43.04)	(35.39, 44.73)

6.4 Correlation between Distributed Models

Based on the results presented in Tables 1 – 3 it is seen that the analytical steady state time in system value (39.83) falls within the 95% CI for all the three random number cases. It not obvious to say which method gives the accurate results. Now, in the tandem queue scenario considered, the inter-arrival times and services times are IID. Hence, for an entity the time spent in downstream servers should be independent (i.e. correlation negligible) of the time spent in upstream servers. The correlation between the time spent in Model A and time spent in Model B observations has been computed, as shown in Table 4.

Table 4: Correlation of Time in Model A and Time in Model B

Repliations	Default stream	Dedicated streams	Separate streams
1	0.345	0.087	0.090
2	0.425	0.089	0.229
3	0.326	0.069	0.059
4	0.396	-0.009	0.126
5	0.332	0.111	0.061

In the case of default streams, the average correlation is found to be 0.365, which is quite large. Since Model A and Model B uses the same default streams, a large service time in Model A for an entity could imply a large service time in Model B, thus increasing their time spent in system. Thus the use of default streams induces correlation in the output measure. This can be expected to increase the overall mean time in system, which is confirmed by the results shown in Table 1. In the case of dedicated streams, the average correlation (0.069) between time spent in Model A and time spent in Model B is found to be negligi-

ble. In the case of separate streams, the average correlation is 0.113.

The surest way of determining the effect of random numbers on the simulation output is to analyze the random numbers used by the distributed simulation models. In the case of COTS packages (viz., ArenaTM) determining the exact sequence of random numbers used during a simulation run is not straightforward. However, in the tandem queueing scenario considered, all its input distributions are exponential. It is known that exponential random variates are generated using inverse transform method (Law and Kelton 2000), with one-to-one mapping of random numbers and the variates generated. In the current work, a partial set of random numbers used in Model A (for generating inter-arrival times) and those used Model B (for generating service times for server₂) are determined. Based on the observations from 5 replications it is found that the average correlation among the partial set of random number used in Model A and Model B are (i) 0.571 for default stream, (ii) 0.022 for dedicated stream and (iii) -0.0009 for separate streams. This clearly illustrates that the use of default random number streams result in correlated output data in distributed simulation.

7 CONCLUSIONS AND FUTURE RESEARCH

To measure the system performance in distributed simulation, a straightforward method will be to separately measure the statistics for each of the distributed models, and then sum these measures to obtain the system performance. Preliminary investigations using a simplistic queueing scenario also revealed the importance of the appropriate use of random number in distributed simulation. Use of default random number streams results in correlated data between the different models of distributed simulation, and hence classical statistical methods cannot be used on those results. Typically dedicated streams of random numbers are assigned to each input distribution over all models. This may be difficult to implement since the number of stochastic variables in the system may be quite large, and requires a central coordination mechanism which may not be possible in distributed setting. A tradeoff between the two is suggested (i.e. use of separate streams) where it is sufficient to ensure that the distributed models use non-overlapping streams of random numbers.

Work is currently being carried out to validate the output analysis methodology for more complex distributed simulations involving resource sharing, multiple entities routings, shared buffer/ conveyor, etc. Also, valid output analysis of non-terminating or steady state distributed simulations pose interesting challenges. Work is underway to develop add-in modules for COTS packages to support and speed up distributed simulation output analysis. The effect of multiple COTS packages interfacing in distributed simulation and their impact on output analysis is also to be

investigated. Eventually, this work is expected to lead to general scheme for output analysis of terminating and non-terminating distributed simulations.

REFERENCES

- Cao Y, X. Jin and Z. Li. 2007. A distributed simulation system and its application. *Simulation Modelling Practice and Theory* 15(1): 21–31.
- Garg, A., J. Venkateswaran and Y.J. Son. 2009. Generic interface specifications for integrating distributed discrete event simulation models. *Journal of Simulation*. (proof)
- Kremien, O., J. Kramer and M. Kapelevich. 1989. Rigorous analysis of (Distributed) simulation results. Available via <http://citeseer.ist.psu.edu/92829.html>
- Law, A.M. and W.D. Kelton. 2000. *Simulation Modeling and Analysis*, 3rd ed. New York: McGraw-Hill.
- Mustafee, N., S.J.E. Taylor, K. Katsaliaki, S. Brailsford. 2006. Distributed simulation with COTS simulation packages: A case study in health care supply chain simulation. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L.F. Perrone, F.P. Wieland, J. Liu, B.G. Lawson, D.M. Nicol and R.M. Fujimoto 1136-1142. Available via www.informs-cs.org/wsc00papers/prog06.htm.
- McLean, C. and F. Riddick. 2000. The IMS MISSION Architecture of Distributed Manufacturing Simulation. In *Proceedings of 2000 Winter Simulation Conference*, ed. J.A. Joines, R.R. Barton, K. Kang and P.A. Fishwick, 1539-1548. Available via www.informs-cs.org/wsc00papers/prog00.htm.
- Taylor, S.J.E. 2003. HLA-CSPIF: The high level architecture COTS simulation package interoperability forum. In *Proceedings of the 2003 Simulation Interoperability Workshop*. Orlando, Florida, USA.

AUTHOR BIOGRAPHIES

B. VINOD KUMAR REDDY is currently a Masters' student of Industrial Engineering and Operations Research at Indian Institute of Technology Bombay. His e-mail address is vinodreddy@iitb.ac.in.

JAYENDRAN VENKATESWARAN is currently an assistant professor in Industrial Engineering and Operations Research at Indian Institute of Technology Bombay. He received his M.S. and Ph.D degrees in Systems and Industrial Engineering from The University of Arizona. His research interest lies in hybrid and distributed simulations, system dynamics methodology, and distributed decision making within integrated supply chains. He is a professional member of IIE, ORSI, SOM and SDS. His e-mail address is jayendran@iitb.ac.in.