A Framework for Analyzing Parallel Simulation Performance

Yong-Meng Teo, Hong Wang and Seng-Chuan Tay
School of Computing
National University of Singapore
Lower Kent Ridge Road
Singapore 119260
email: teoym@comp.nus.edu.sg

Abstract

This paper provides a framework for studying the complex performance interactions in parallel simulation along three main components: simulation model, parallel simulation strategy/protocol, *and* execution platform. *We propose a methodology for characterizing the potential parallelism of a simulation model based on analytical modeling techniques. A clear understanding of the degree of event parallelism inherent in the simulation problem/model is essential for the simulation practitioners to assess the performance benefits of exploiting parallelism before substantial programming effort is invested in its implementation. Establishing the baseline event parallelism available in a simulation model is crucial to the simulation practitioners for assessing the performance (parallelism) loss that may arise from the parallel synchronization protocol, and the architecture of the parallel execution platform used. We analyze how causality dependency of event affects the performance of simulation model, and determine the potential event parallelism in simulation models.*

1 Introduction

Parallel discrete-event simulation (PDES) research in the past two decades have focussed mainly on the pioneering works by Chandy and Misra [1] and Jefferson [4]. Based on the concept of logical process (LP) and the virtual time paradigm, the two main synchronization protocols are *conservative* and *optimistic* [3]. A synchronization protocol ensures that causally related events are processed in timestamped order. For a *conservative protocol*, each LP does not proceed with the execution of its next event until it can ascertain that future message with a smaller timestamp will not occur. In the *optimistic protocol*, each LP executes events regardless of the possibility of receiving messages in their logical past. Performance analysis of parallel simulation is a hard but indispensible issue that must be addressed. It is important to predict the performance of parallel simulation before a simulationist invests a substantial effort in developing parallel codes. If the inherent parallelism of the simulation application is not sufficiently large, it is not worthwhile to take the parallel simulation approach [8].

Performance analysis of parallel simulation is critical to the future success and general acceptance of parallel simulation in practice. Many experimental studies on the performance of parallel simulation protocols using metrics such as speedup and the number of events executed per unit time have been reported [2, 11]. Although these metrics are helpful in evaluating the usefulness of a particular simulation protocol on a given execution platform, these studies do not reveal whether there is sufficient parallelism in the simulation applications to be exploited before considering sophisticated parallel simulation protocols and parallel execution platforms.

This paper presents a hierarchical framework for studying parallel simulation performance from simulation model to its implementation. Section 2 presents an overview of related work. Section 3 discusses the main performance interaction components in a parallel simulation. In section 4, we present our performance analysis methodology for quantifying the degree of available parallelism in a simulation model. We derive the parallelism of a simulation model using a divide-and-conquer approach: (a) without considering causality effect of events by applying operational analysis, and (b) with causality effect of events using stochastic analysis. This approach is useful in analyzing large and complex simulation systems with different degrees of performance accuracy. Section 5 contains our concluding remarks.

2 Related Work

The notion of critical path [10] has long been used in analyzing the performance of parallel simulation. The *critical path length* for a simulation is calculated from the real time

0-7695-0128-1/99 $10.00 © 1999 IEEE

it takes to execute each event, and from the precedences induced among the events by causal dependencies. Research in critical path analysis can be classified into two major areas: *critical time* and *super-criticality*.

For conservative mechanism, the highest critical time of any event in a simulation provides a lower bound of the time it takes to execute a simulation. Critical time can be computed in two ways: using *events trace* or *space-time diagram* [10, 12], or by *direct instrumentation* of the simulation program [7]. However, critical time cannot generally provide a lower bound for optimistic simulation because some optimistic protocols exhibit super-critical speedup[5, 14]. There are a number of drawbacks in using critical path analysis (CPA). Firstly, CPA assumes that an implementation exist for gathering the necessary information to construct the acyclic dependency graph. This may not be practical when applying the analysis to predict the performance of a simulation application before it is developed. However, CPA may be a suitable tool for tuning existing simulators. Secondly, CPA uses the run time of events collected from a sequential simulation run to approximate its event critical time. This may not be representative for a parallel simulation.

At the preliminary stage of a simulation study, it is practical to assume that a simulator does not exist for taking measurements as required by some performing analysis approaches. Wagner and Lazowska [15] views a PDES application as a queueing network. To determine the performance potential for a given simulation model, they proposed an analytical method that is based on the summation of normalized utilization of independent LPs. The *normalized utilization* of an LP is obtained by setting the largest LP utilization to 1 and changing other LP utilizations proportionally. This method is used to obtain an asymptotic upper bound of parallelism in a simulation model. However, when analyzing the performance of a simulation model, we are also concerned with its performance for different workloads. Thus, the asymptotic upper bound on simulation model parallelism provides an estimate that is too coarse for most practical cases.

3 Performance Interactions

We divide the *performance interactions* in a parallel simulation into *three* main components. The *simulation model* views the system to be simulated as a queueing network of logical processes (LPs) that communicates by exchanging timestamped messages or events. Each LP models an entity in the simulation problem. This is the typical simulation model adopted in parallel simulation [4]. The *parallel simulation strategy* implements the synchronization protocol to ensure that each LP executes events in the timestamp order. The computational speed of processors, communication topology and latency, etc. in the *execution platform* can influence the simulation performance. The main aim of this paper is to predict the inherent parallelism in a simulation model before its implementation. This is crucial in assessing the suitability of applying parallel simulation technology before committing resources for its implementation.

Table 1 contains a list of performance parameters used in this paper. In parallel processing, typically a parallel algorithm/program is analyzed to determine its degree of available parallelism. A common measure of program parallelism (π) is defined as

$$\pi = \frac{T_1}{T_\infty} \tag{1}$$

where T_1 and T_∞ are the total time required to execute the program on one processor and on an infinite number of processors, respectively [6].

In the context of parallel simulation, a simulation problem can be described as a simulation model and event as the unit of work. Therefore, we can define the parallelism of a simulation model, π_{model} as follows,

$$\pi_{model} = \frac{T_1}{T_n} \tag{2}$$

where T_1 is the total time taken required to execute the simulation application sequentially, and T_n is the total time required to execute the simulation model consisting of n LPs on n processors. To exploit maximum simulation model parallelism, we assume that each LP is mapped onto one processor for execution.

If W_i denote the total event execution time on LP_i, then

$$T_1 = \sum_{i=1}^{n} W_i$$

Substituting T_1 in the equation (2), we have

$$\pi_{model} = \sum_{i=1}^{n} \frac{W_i}{T_n}$$

Clearly, $\frac{W_i}{T_n}$ is the utilization for LP_i. Therefore,

$$\pi_{model} = \sum_{i=1}^{n} U_i \tag{3}$$

where U_i denotes the utilization of LP_i, and π_{model} denotes the parallelism for a simulation model consisting of n LPs. Thus, the parallelism of a simulation model can be defined as the sum of all LP utilizations in the network. This result is similar to that proposed by Wagner and Lazowska for describing the parallelism of PDES applications [15].

In a *simulation problem*, let the job arrival rate be λ, the service rate of the server be μ. The utilization of the server is $U = \frac{\lambda}{\mu}$. For a system with k servers, the

PARAMETER			DESCRIPTION
simulation problem		p_i	routing probability to the ith server
		λ_i	mean arrival rate of jobs to the ith server (open system)
		λ	sum of λ_i for $\forall$ servers
		n	number of jobs (closed system)
simulation model		σ_i	mean execution time of an event at the ith LP
		ω_i	mean message arrival rate at the ith LP
derived	causality effect free	U_i	utilization of the ith LP
		X	event throughput of the simulation model (open system)
		$\pi_{model'}$	parallelism of the simulation model (open system)
		$X(n,p)$	event throughput of the simulation model (closed system)
		$\pi_{model'}(n,p)$	parallelism of the simulation model (closed system)
	causality effect	ρ	causality factor of simulator
		π'_{model}	parallelism of the simulation model with causality effect

Table 1. Performance Model Parameters

parallelism in the simulation problem ($\pi_{problem}$) is $\frac{k\lambda}{\mu}$. For a *simulation model*, let the message arrival rate at each LP be ω, and the LP message processing rate be σ, then the LP utilization is $\omega\sigma$. For k LPs, the simulation model parallelism (π_{model}) is $k\omega\sigma$. Since λ and μ are application related parameters, and ω and σ are simulation model parameters, the parallelism in a simulation problem and that of its simulation model may be significantly different and their relationship has to be established.

4 Parallelism Analysis Methodology

Based on the performance interaction components discussed, a hierarchical parallelism analysis methodology is depicted in figure 1. This paper focusses on characterizing

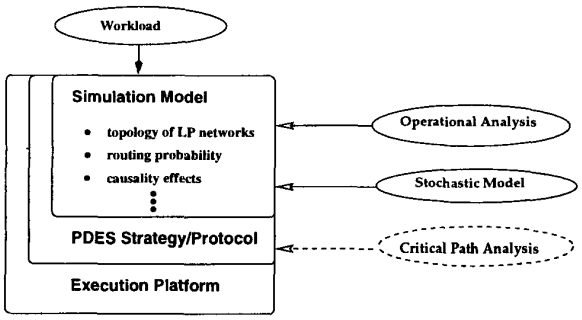


Figure 1. Parallelism Analysis Methodology

the parallelism of a simulation model. This metric is independent of synchronization protocol and execution platform. We consider the effects of LP topology, message routing probability and causal dependency of events on simulation model parallelism. Extension of this approach to include performance loss due to the PDES strategy and the execution platform is beyond the scope of this paper.

Definition 1. A simulation problem can be viewed as a graph $G(V, E)$, where the vertex V denotes the servers and the edge E denotes the connection between servers. A **simulation problem** is represented by a function $P(\vec{\lambda}, \vec{\mu}, G)$, where the vector $\vec{\lambda}$ denotes the arrival rates at servers, and the vector $\vec{\mu}$ denotes the service rates.

Definition 2. The **simulation model** for a given simulation problem is represented by a function $M(\vec{\lambda}, \vec{\mu}, G, \vec{\omega}, \vec{\sigma})$. The vectors $\vec{\omega}$ and $\vec{\sigma}$ are simulation model parameters that denote the message arrival rates and the event execution times at LPs respectively.

To characterize the parallelism of a simulation model and the causality effect of events, we partition a graph $G(V, E)$ into a set of subgraphs, S, where $S = S_1, \ldots, S_z$,

$$G = \bigcup_{i=1}^{z} S_i$$

and each subgraph is separately analyzed. We summarize the steps to characterize simulation model parallelism as follow:

- **Step 1.** Establish the simulation model for a given simulation problem, i.e. $M(\vec{\lambda}, \vec{\mu}, G, \vec{\omega}, \vec{\sigma})$.
- **Step 2.** Consider a simplified simulation model M called M', where M' is a function of G, $\vec{\omega}$, and $\vec{\sigma}$, and evaluate the simplified model $M'(G, \vec{\omega}, \vec{\sigma})$ analytically to obtain the model parallelism $\pi_{model'}$.
- **Step 3.** Partition the model graph G into a set of subgraphs S. For each $S_i \in S$, analyze the causality factor (ρ_i) with respect to the parameters of $\vec{\lambda}, \vec{\omega}$, and S_i. The *causality factor* (ρ) quantifies the parallelism loss due to causal dependency of events in a simulation model.
- **Step 4.** Determine the total causality factor for the simulation model, i.e. $\rho = f(\rho_1, \ldots, \rho_z)$, where f

is a function of the partitioning scheme. The method to determine the function f is left as future work.

- **Step 5.** Compute the parallelism for a simulation model, $M(\vec{\lambda}, \vec{\mu}, G, \vec{\omega}, \vec{\sigma})$ as follows:

$$\pi'_{model} = \pi_{model'} \times \rho$$

We analyze simulation model parallelism without and with causality effect separately.

4.1 Parallelism Analysis without Causality Effect

This section shows how to derive the parallelism of a simplified simulation model $M_1(G, \vec{\omega}, \vec{\sigma})$, i.e. without considering causality effect. Figure 2 shows an abstraction of an LP consisting of an input queue, an output queue, and an event execution unit (EEU). The EEU can be viewed as

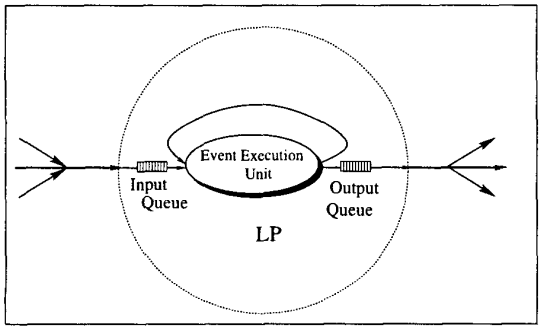


Figure 2. A Logical Process

a sequential discrete event simulator. We define *LP utilization* as the ratio of time that a EEU is busy processing event over the simulation duration. Since EEU executes events sequentially and at most one event can be processed by an EEU at a time, its utilization cannot exceed 1. We consider the effects of LP network topology and message routing probability that influence the parallelism in simulation model.

4.1.1 Topology of Simulation Model and Parallelism

For simplicity of analysis, we divide a simulation model into three basic LP topologies: serial, parallel, and hybrid (see figure 3). First, we assume that causality error due to message mis-orders at LPs will not affect the simulation results. This assumption is valid when the PDES exhibits supercritical speedup, and provides a performance bound for applications where supercriticality exists.

Suppose message flows are uni-directional. In an open system where the event arrival rate is greater than the service rate, event messages are accumulated in the input queue of LPs. When input queue is full, arriving messages may be lost. In our analysis, we assume balanced message flow, i.e. event service rate is equal or faster than the message arrival rate. Applying operational analysis to a *serially connected LP* network as shown in figure 3(a), the utilization of each LP based on the *utilization law* [9] is

$$U_i = X\sigma_i \tag{4}$$

where throughput (X) is defined as the number of events executed per second, and σ_i is the mean event execution time on LP_i. Substituting equation (4) to equation (3), we have

$$\pi_{model'} = \sum_{i=1}^{n} X\sigma_i = X\sum_{i=1}^{n} \sigma_i \tag{5}$$

For a *parallel LP topology* shown in figure 3(b), the utilization of each LP is computed as follows:

$$U_i = p_i X\sigma_i \tag{6}$$

where p_i is the message routing probability to LP_i. Using equation (3), we have

$$\pi_{model'} = \sum_{i=1}^{n} p_i X\sigma_i = X\sum_{i=1}^{n} p_i\sigma_i \tag{7}$$

A *hybrid LP topology* (see figure 3(c)) combines the serial and parallel LP networks. Suppose each branch i consists of m_i LPs connected linearly and $\pi_{model',i}$ denotes its parallelism. From equation (5), we have

$$\pi_{model',i} = \sum_{j=1}^{m_i} p_i X\sigma_{i,j} = p_i X\sum_{j=1}^{m_i} \sigma_{i,j} \tag{8}$$

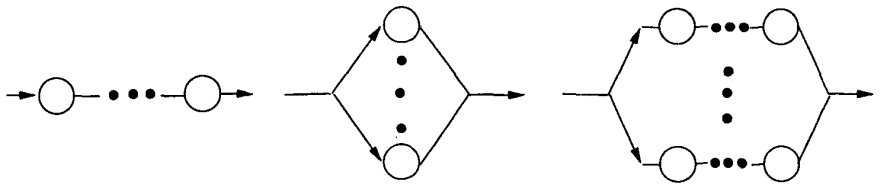
For the whole network, by applying (7), we get,

$$\pi_{model'} = \sum_{i=1}^{n} \pi_i = \sum_{i=1}^{n} p_i X\sum_{j=1}^{m_i} \sigma_{i,j} = X\sum_{i=1}^{n} p_i \sum_{j=1}^{m_i} \sigma_{i,j} \tag{9}$$

Equation (9) shows that the parallelism of an LP model is a *linear* function of throughput. In summary, to analyze an LP network for a given workload we determine the throughput of the system, and the utilization of each LP. Summing up all LP utilizations gives the parallelism of the simulation model without causality effect. In the next section, we apply the *Mean Value Analysis (MVA)* [9] to compute the throughput for a closed LP network.

4.1.2 Effects of Routing Probability

Figure 4 shows a closed queueing network consisting of three LPs with one fork point and one merge point. Let σ_i denote an exponentially distributed random variable that models the execution time for an event at LP_i. For ease of analysis, we assume that $\sigma_1 = \sigma_2 = \sigma_3 = \sigma$. Let p and



(a) Serial LPs (b) Parallel LPs (c) Hybrid LPs

Figure 3. Basic LP Topologies

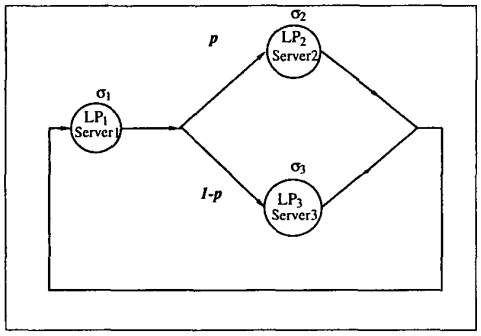


Figure 4. An LP Network with Three Servers

$(1-p)$ denote the routing probabilities from LP_1 to LP_2, and LP_1 to LP_3 respectively. Let n denote the number of events executed in the LP network. Using equation (9), we derive $\pi_{model'}(n,p)$ for $p \in [0,1]$ as follows:

$$\pi_{model'}(1,p) = 1$$

$$\pi_{model'}(2,p) = 2 \times \frac{2}{3-p+p^2}$$

$$\pi_{model'}(3,p) = 2 \times \frac{3-p+p^2}{4-3p+3p^2}$$

$$\pi_{model'}(4,p) = 2 \times \frac{4-3p+3p^2}{5-6p+7p^2-2p^3+p^4}$$

$$\pi_{model'}(5,p) = 2 \times \frac{5-6p+7p^2-2p^3+p^4}{6-10p+14p^2-8p^3+4p^4}$$

$$\pi_{model'}(6,p) = 2\times \frac{6-10p+14p^2-8p^3+4p^4}{7-15p+25p^2-21p^3+13p^4-3p^5+p^6}$$

$$\pi_{model'}(7,p) = 2\times \frac{7-15p+25p^2-21p^3+13p^4-3p^5+p^6}{8-21p+41p^2-45p^3+35p^4-15p^5+5p^6}$$

$$\pi_{model'}(8,p) = 2\times \frac{8-21p+41p^2-45p^3+35p^4-15p^5+5p^6}{9-28p+63p^2-85p^3+81p^4-49p^5+21p^6-4p^7+p^8}$$

Figure 5 shows that the simulation model parallelism

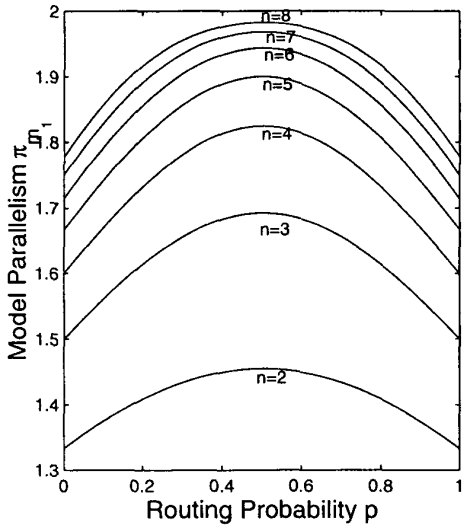


Figure 5. Model Parallelism ($\pi_{model'}$) and Routing Probability (p)

$(\pi_{model'}(n,p))$ increases as workload (n) increases. The coefficient of each term is generalized as follows:

constant : $a_n^0 = n+1$,
p item : $a_n^1 = -(C_n^2)$,
p^2 item : $a_n^2 = C_n^3 + a_{n-1}^0$,
p^3 item : $a_n^3 = -(C_n^4 + a_{n-2}^1)$,
p^4 item : $a_n^4 = C_n^5 + a_{n-3}^2$,
where C_n^i represents the combination coefficient $\binom{n}{i}$.

The coefficients of a p^i term, for $i \leq n-1$, can be represented by the following recursive equation:

$$a_n^i = (-1)^i (C_n^{i+1} + a_{n+1-i}^{i-2})$$

When $n = 2k+2$ and k is a non-negative integer

$$a_n^n = 1$$

Let

$$A_n(p) = a_n^0 + a_n^1 p + \ldots + a_n^{n-1} p^{n-1}$$

when n is odd number, and

$$A_n(p) = a_n^0 + a_n^1 p + \ldots + a_n^{n-1} p^{n-1} + p^n$$

when n is even number.

The generalized equation for simulation model parallelism can be expressed as follows:

$$\pi_{model'}(n,p) = 2\frac{A_{n-1}(p)}{A_n(p)} \tag{10}$$

It follows from equation (10) that $\pi_{model'}(n,p) \geq \frac{2n}{n+1}$, and $\pi_{model'}(n,p) \leq 2$. Therefore, as $n \to \infty$, $\pi_{m_{model'},\infty} \to 2$.

For a generalized network as depicted in figure 6, and

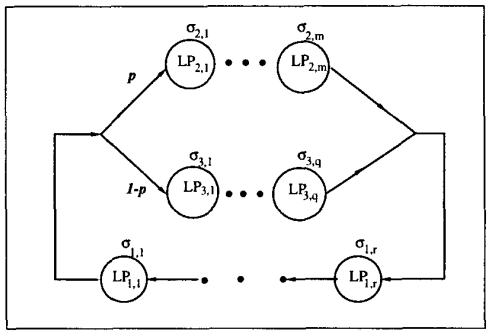


Figure 6. A Generalized LP Network

using equation (9), we have

$$\pi_{model'}(n,p) = \sum_{i=1}^{r} U_{1,i} + \sum_{j=1}^{m} U_{2,j} + \sum_{k=1}^{q} U_{3,k}$$
$$= rX(n,p)\sigma + mpX(n,p)\sigma + q(1-p)X(n,p)\sigma$$
$$= [r + q + (m-q)p]X(n,p)\sigma \tag{11}$$

Equation (11) shows that routing probability affects the simulation model parallelism. Since $X(n,p)\sigma$ represents the utilization of $LP_{1,i}$, for $i = 1 \ldots r$, we have,

$$X(n,q)\sigma \leq 1 \tag{12}$$

Therefore, $\pi_{model'}(n,p) \leq [r + q + (m-q)p]$ will always hold, and the throughput

$$X(n,p) \leq \frac{1}{\sigma} \tag{13}$$

However, for different workloads and routing probabilities the system will produce different throughputs. Given parameters r, m, and q, the throughput can be computed by Mean Value Analysis.

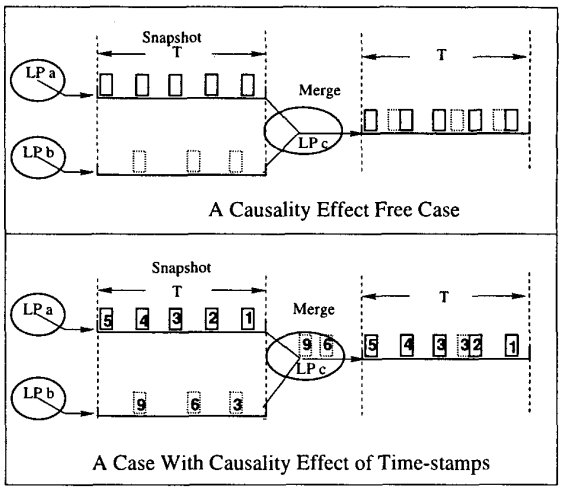


Figure 7. An Example of Causality Effect

4.2 Parallelism Analysis with Causality Effect

Consider two time-stamped message streams (from LP_a and LP_b) that arrive at LP_c (see figure 7). Within each stream the messages are in time-stamp order, and the messages must also be processed in time-stamp order by LP_c. We first assume that the two message arrival streams from independent Poisson processes, and that the successive time-stamps in each stream are independent sequences of Poisson epochs. Consider a snapshot of duration T, LP_a produces a stream of five messages and LP_b a stream of three messages that are routed to LP_c for execution. In the case without considering causality effect, all eight messages are executed by LP_c within the period T (see figure 7). If causality effect is considered, messages must be executed according to the time-stamp order resulting in a lower throughput of six messages. This is similar to the idealized conservative synchronization scheme without null messages. To maintain event causality, a message with a higher time-stamp that arrives earlier is delayed for execution until all other messages with lower time-stamps arrive.

4.2.1 Feed-Forward Open LP Networks

Shorey and Kumar [13] presented an approach to compute the message throughput for a simulator using a feed-forward open queueing network example as shown in figure 8. The queueing model is mapped onto the distributed simulator with each LP simulating a server. Consider k Poisson processes that model the message arrival streams with rates ω_i, for $1 \leq i \leq k$, and that the successive time-stamps in each stream are Poisson epochs with rates λ_i. Assume $\omega_i/\lambda_i \neq \omega_j/\lambda_j$, *where* $i \neq j$. Let $j^* = \min_{1 \leq i \leq k}(\omega_i/\lambda_i)$. The departure process at LP_{k+1} com-

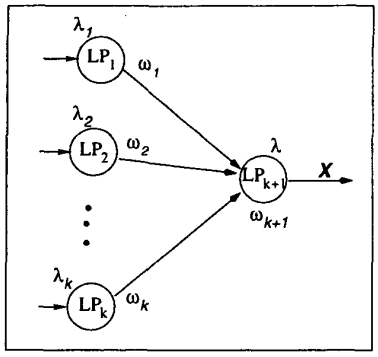


Figure 8. A Feed Forward LP Network with Merging

prises geometrically distributed flows of messages departing at the arrival epochs of stream j^*. The message departure epochs form a renewal process of rate $\omega_{j^*} \sum_{i=1}^{k} \lambda_i$, and the time-stamp process is Poisson with rate $\sum_{j=1}^{k} \lambda_i$. Such a stream is denoted by $(\omega_{j^*}, \lambda_{j^*}, \lambda)$, where $\lambda = \sum_{i=1}^{k} \lambda_i$.

Theorem 1 (Shorey & Kumar [13]). *If the processors of the LPs at the non-leaf nodes have an infinite service rate, then the departure process of the simulator is* $(\omega_{j^*}, \lambda_{j^*}, \lambda)$ *where* $j^* = arg\ \min_{1 \le j \le k}(\omega_j/\lambda_j)$, *and k is the number of leaf nodes. The throughput of the simulation model with causality effect taken into consideration is*

$$X' = (\min_{j=1}^{k} \omega_j/\lambda_j) \times \lambda \tag{14}$$

Assuming message flow balance and ignoring causality effect, the throughput is

$$X = \sum_{j=1}^{k} \omega_j \tag{15}$$

Next, we define the causality factor ρ as

$$\rho = \frac{X'}{X} = \frac{(\min_{j=1}^{k} \omega_j/\lambda_j) \times \lambda}{\sum_{j=1}^{k} \omega_j}$$

where $\rho \le 1$.

Let π'_{model} denote the simulation model parallelism when we consider the causality effects of message time-stamps. The parallelism relationship between π'_{model} and $\pi_{model'}$ is expressed as follows:

$$\pi'_{model} = \rho \times \pi_{model'} \tag{16}$$

Discussion 1.

Assume k homogeneous LPs with message arrival rates $\omega_1 = \omega_2 = \ldots \omega_k$, event execution time $\sigma_1 = \sigma_2 = \ldots \sigma_k$, and job arrival rates in the simulation problem $\lambda_1 = \lambda_2 = \ldots = \lambda_k$. We have

$$\rho_k = \frac{(\min_{j=1}^{k} \omega_j/\lambda_j) \times \sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{k} \omega_j} = \frac{\omega_j/\lambda_j \times \lambda_j \times k}{k \times \omega_j} = 1$$

and

$$\pi'_{model} = 1 \times \sum_{j=1}^{k} (\omega_j \times \sigma_j) + \sum_{j=1}^{k} \omega_j \times \sigma_{k+1} = k \times \omega_1 \times (\sigma_1 + \sigma_{k+1}) \tag{17}$$

Therefore, the model parallelism is linearly related to the number of LPs before the merge point.

Discussion 2.

To vary the workload in the simulation problem, we change the job arrival rates λ_j. Based on the same assumptions in "discussion 1", thus $\rho_k = 1$. Therefore, the model parallelism remains constant, i.e. model parallelism is independent of the job arrival rates in the problem.

Discussion 3.

Varying the messsage arrival rates ω_i, the model parallelism is linearly related to the message arrival rates. This observation is valid when the message arrival rate is less than the event execution rates of LPs, i.e. $\omega_j \le 1/\sigma_j$.

4.2.2 LP Network with Split-then-Merge

We replace the k Poisson job generator processes with a single Poisson process with rate λ as shown in figure 9. The

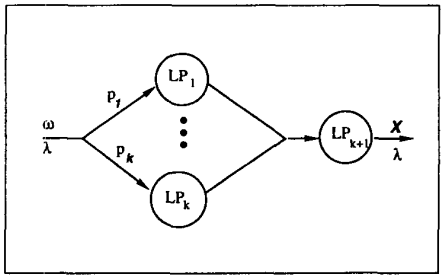


Figure 9. An LP Network with Split-then-merge

message arrival rate (ω) is splitted to feed k LPs with equal probability. Messages that depart from these k LPs are merged as they arrived at LP_{k+1}. Assuming a job flow balance system, we show that message throughput of LP_{k+1} is equal to ω.

Since the message flow at splitting point is ω, the message flow to feed LP_i will be p_i. Assume that job flow at

the same point has the same splitting phenomenon. The job flow to feed the server which is represented by LP_i is $p_i\lambda$. Since the LP model is job flow balanced, we assume that the service rate at LP_i $(for i = 1 \ldots k)$ is large enough, the message throughputs of LP_i is ωp_i. The job throughput of each server in the simulated system, which corresponds to LP_i is λp_i. Therefore, we get k message flows merged to feed LP_{k+1}. According to theorem 1, the throughput for LP_{k+1} is

$$\begin{aligned} X &= \min_{i=1}^{k} \left\{ \frac{\omega p_i}{\lambda p_i} \right\} \times \lambda \\ &= \{\omega/\lambda\} \times \lambda = \omega \end{aligned} \quad (18)$$

This implies that the message throughput of LP_{k+1} is the same as the message arrival rate at the splitting point. In this case we have $\rho = 1$, i.e. the causality effect of of message time-stamps does not influence the event parallelism of the LPs network.

5 Conclusions

We focus on characterizing simulation model parallelism to provide simulation practitioners with an estimate of the performance potential of a simulation application before substantial efforts are invested in its implementation. The proposed parallelism metric is independent of synchronization protocol and execution platform, and measures the event parallelism inherent in the simulation model. It provides a useful upper bound on exploitable event parallelism for assessing the performance loss due to synchronization protocol and the execution platform used. For example, if the synchronization protocol used does not exploit all the inherent parallelism in the model, the exploited event parallelism will be smaller, and their difference indicates the performance improvement achievable.

The analysis of parallelism potential of PDES model was carried out in two steps. We simplify the model by ignoring the causality effect. By applying classical performance analysis such as operational laws and the mean value analysis, we obtain an upper parallelism bound. This bound is also applicable to PDES applications that exhibit supercritial speedup. Next, we apply the results of open queueing network from [13] to show how the causality effects of time-stamps influence the parallelism of the model. We show that in a LP network with a split-then-merge topology, causality effect does not have a significant impact on model parallelism. Extension of this parallelism framework to include sychronization protocol and execution platform are discussed in a separate paper.

Acknowledgement

This research is supported by a grant from the Ministry of Education and the Port Authority of Singapore under grant RP960715.

References

[1] K.M. Chandy and J. Misra, *"Distributed Simulation: A Case Study in Design and Verification of Distributed Programs,"* IEEE Transactions on Software Engineering, SE-5, 5, pp. 440-452, September 1979.

[2] A. Ferscha, J Johnson and S.J. Turner, *"Early Performance Prediction of Parallel Simulation Protocol,"* Proceedings of the World Conference on Systems Simulation'97, pp. 282–287, September, 1997.

[3] R.M. Fujimoto, *"Parallel Discrete Event Simulation,"* Communication of ACM, 33(10), pp. 30-53, 1990.

[4] D.A. Jefferson, *"Virtual Time,"* ACM Transactions on Programming Languages and Systems, Vol. 7, No. 3, pp. 404-425, July 1985.

[5] D. Jefferson and P. Reiher, *"Supercritical Speedup,"* Proceedings of the 24th Annual Simulation Symposium, 1991.

[6] C.E. Leiserson and H. Prokop, *"A Minicourse on Multithreaded Programming,"* http://theory.lcs.mit.edu/ cilk, 1998.

[7] Y.B. Lin, *"Parallelism Analyzers for Parallel Discrete Event Simulation,"* ACM Transactions on Modeling and Computer Simulation, 2(3), pp. 239-264, 1992.

[8] Y.B. Lin, *"Will Parallel Simulation Research Survive,"* ORSA Journal of Computing, Vol. 5, No. 3, pp. 236-238, 1993.

[9] D.A. Menasce, et. al, *"Capacity Planning and Performance Modeling"*, Prentice-Hall, 1994.

[10] B. J. Overeinder and P. M.A. Sloot, *"Parallel Performance Evaluation Through Critical Path Analysis, "* Lecture Notes of Computer Science, Vol. 919, 1995.

[11] G.D. Peterson and R.D. Chamberlain, *"Performance of a Globally-Clocked Parallel Simulator,"* International Conference on Parallel Processing, 1993.

[12] M. Salmi, H. Harju, and J. Porras, *"Computing the Parallel Potential of Event-oriented Simulation Applications,"* Proceedings of the European Simulation Symposium 1994, 1994.

[13] R. Shorey, A. Kumar, and K.M. Rege, *"Instability and Performance Limits of Distributed Simulators of Feedforward Queueing Networks,"* ACM TOMACS, 7(2), pp. 210-238, 1997.

[14] S. Srinivasan and P.F. Reynods, *"Super-Criticality Revisited,"* Proceedings of 9th Workshop on Parallel and Distributed Simulation, 1995.

[15] D.B. Wagner and E.D. Lazowska, *"Parallel Simulation of Queuing Networks : Limitations and Potentials,"* ACM Performance Evaluation Review, 17(1), pp. 146-155, 1989.