Massively Parallel Simulations with Application to Queueing Networks

Jain J. Wang and Marc Abrams

February 15, 1995

Abstract

This paper investigates two parallel simulation methodologies, *multiple replication* and *parallel regenerative simulation*. Problems of applying these methodologies to massively parallel simulations are identified. Two approaches (MR-PI and PR) are proposed to overcome some of these problems. The MR-PI approach, based on multiple replication simulation, uses a pilot simulation to reduce the initial transient bias by starting the simulation from a state that is representative of the steady-state conditions. The PR approach allows approximate regeneration by using substate matching for models that do not regenerate frequently. Empirical results suggest that the proposed approaches can produce very accurate results.

1 Introduction

Simulation has been widely used as a tool to study system performance in communication networks and many other areas. The major drawback of simulation, however, is that it is computationally intensive. Simulating a complex system or rare events in a simple system may require days, or even weeks of computer execution time. With the advent of massively parallel computers, where hundreds or even thousands of processors are connected together in a single computer to provide up to teraflop computing power, parallel simulation can potentially reduce simulation execution time by several orders of magnitude. A major challenge in running simulations on a massively parallel computer is to exploit sufficient parallelism for a large number of processors.

Many parallel simulation algorithms proposed previously are *space-parallel* (see [6] for a survey), in which parallelism is obtained through a partitioning the simulation model into components in the space domain of the model. For a queueing network model, for instance, the model can be partitioned into a number of queues and each processor may simulate a single queue concurrently with proper synchronization. Parallelism provided by space-parallel simulation is limited by the number of components that exist in the simulation model (e.g., the number of queues in the target queueing network model).

Two alternative methodologies are *multiple replication* simulation and *parallel regenerative* simulation. In this paper, we investigate these two methodologies and propose some approaches suitable for massively parallel simulation. The proposed techniques and the related methodologies are discussed in Sections 2 and 4. Section 6 evaluates the proposed approaches experimentally with some queueing systems suitable for modeling communication networks. Conclusions are given in Section 7.

Problem Statement

The time evolution of a simulation model can be viewed as a realization of a random process $\mathbf{X} = \{X(t), t \geq 0\}$. Assume that $X(t)$ converges in distribution to some random variable X as $t \to \infty$. We use simulation to estimate the steady-state mean $\mu = E(X)$. The problem of interest is to parallelize the simulation so that a large number of processors can be used to execute the simulation efficiently while producing an accurate estimate for μ.

2 Multiple Replication Simulation

In this section, we first review the multiple replication methodology (Section 2.1) and then, based on this methodology, we present an algorithm that is suitable for massively parallel simulation (Section 3.1).

2.1 Multiple Replication Simulation Primer

An intuitive parallel simulation approach is to make each processor simulate a replication of the target simulation model independently beginning with a fixed state for T simulation time units. Let P denote the number of processors. Suppose processor $i, 1 \leq i \leq P$, generates M_i *observations* $Y_{i1}, Y_{i2}, \ldots Y_{iM_i}$, where $0 = t_{i0} < t_{i1} < \ldots < t_{iM_i} = T$ and

$$Y_{ij} = \int_{t_{ij-1}}^{t_{ij}} X(t)dt,$$

for $1 \leq j \leq M_i$. Let

$$\bar{Y}_i = \frac{\sum_{j=1}^{M_i} Y_{ij}}{T}.$$

Then $\bar{Y}_1, \bar{Y}_2, \ldots \bar{Y}_P$ are independent and identically distributed (IID) random variables. An obvious estimator for μ, denoted $\hat{\mu}_1(P,T)$, based on $\bar{Y}_i's$ can be obtained by:

$$\hat{\mu}_1(P,T) = \frac{\sum_{i=1}^{P} \bar{Y}_i}{P}.$$

By the central limit theorem, an approximate $100(1-\alpha)$ percent confidence interval for μ is:

$$\hat{\mu}_1(P,T) \pm z_{1-\alpha/2}\sqrt{\frac{S^2(P)}{P}},$$

where $S^2(P) = \sum_{i=1}^{P} \frac{[\bar{Y}_i - \hat{\mu}_1(P,T)]^2}{P-1}$ is the sample variance of $\bar{Y}_i$, and $z_{1-\alpha/2}$ is the upper $1-\alpha/2$ critical point for the standard normal distribution (i.e., $\mathbf{N}(0,1)$).

Generally, because the simulation does not start from the 'steady-state condition' of the system, the transient period before the system reaches the steady-state will cause a bias in the simulation outcome. Therefore, to achieve a high accuracy, a simulation using estimator $\hat{\mu}_1(P,T)$ requires that T be sufficiently large to alleviate the bias due to the initial transient. This constraint imposes a lower bound on the simulation time regardless of the number of processors used.

3 Multiple Replication Simulation with Deletion

A commonly suggested solution to this problem is to discard the observations collected in an estimated initial transient (or *warm-up*) period $[0, t_w)$, where $0 \leq t_w < T$ is the estimated starting time of steady-state, and $T - t_w$ is the effective simulation length. The multiple replication approach with warm-up deletion will be referred to as the *MR-D approach* throughout our discussion.

For parallel simulation, the use of multiple processors is justified by a shorter simulation completion time or using fewer overall CPU cycles while achieving the same level of accuracy that would have been achieved by a sequential simulation. Therefore, for the MR-D approach, when P increases, the simulation length of each processor must decrease accordingly. When the simulation length of each processor becomes too short, the MR-D approach will fail to remove the initial transient bias. Therefore, it is important that the MR-D simulation starts from a state that is relatively 'close' to the steady-state condition in order to achieve a small initial transient. For queueing systems, Kelton [13] has shown that an initial state that is as congested (in terms of the number of jobs that exist in the system) as the steady-state mean can induce comparatively short transients. However, due to a lack of steady-state information, the deterministic initial state for the MR-D simulation is usually selected *arbitrarily* and thus significant bias may result.

To address this problem, Kelton [13] proposed a random initialization method, which assumes that the set of all possible initial states of the simulation model can be represented by a finite integer set $I_r = \{1, 2 \ldots, r\}$. The random initialization method consists first of one short pilot simulation run that records the 'maximum' state $m \leq r$ that is observed. Then the initial states of the actual production runs are selected randomly from $\{1, 2 \ldots, m\}$. Kelton showed that such stochastic initialization usually results in a smaller bias than a deterministic initialization. However, to apply this method, we still face the problem of initializing the pilot run. Kelton has suggested deterministic initialization for the pilot run. However, how to choose a deterministic initial state (i.e., for the pilot run) is not addressed in [13].

Kelton's random initialization method is not appropriate for massively parallel simulation. This is because when P is large, the pilot simulation run length needs to be short (in order to achieve a short overall simulation completion time) and thus the resultant upper bound m will be relatively close to the initial state of the pilot simulation. Hence, the pilot simulation will effectively be fruitless. In the next section, we present a stochastic initialization approach suitable for parallel simulation.

3.1 Multiple Replication Simulation with Polling Initialization

We now present a pilot simulation algorithm that allows a short run length while yields initial states that are representative of the steady-state. Let $\{X_s(t), t \geq 0\}$ be the random process representing the target simulation model that is initialized with state $s \in I_r$. That is, $X_s(0) = s$ with probability 1, and $X_s(t) \Rightarrow \mu$ as $t \rightarrow \infty$.

Let $\hat{\mu}(s,T) = \frac{1}{T}\int_0^T X_s(t)dt$ be an estimate of μ after a simulation of T time units long that uses s as the initial state. The bias of $\hat{\mu}(s,T)$ is $b(s,T) = E(\hat{\mu}(s,T)) - \mu$. Assume that there exists a small value $T_0 > 0$ such that for all $T \geq T_0$, the absolute amount of bias (i.e., $|b(s,T)|$) decreases monotonically as T increases (alternatively $E(\hat{\mu}(s,T))$ converges monotonically toward μ). Kelton and Law [12] has identified a class of processes that satisfies this assumption with $T_0 = 0$.

Suppose there exists a state $s_{opt} \in I_r$ that is sufficiently representative of the steady-state condition such that $b(s_{opt},T) \leq b(s,T)$ for all s and $T > 0$. Because s_{opt} is generally unknown, our approach uses a pilot simulation that applies a polling initialization (PI) technique to determine a small interval $[a, b]$ (where $a \in I_r, b \in I_r$, and $a \leq b$) that is likely to contain s_{opt}. Then the initial states of the actual production runs are selected randomly from the range $[a, b]$. In the PI pilot simulation, we associate each state s with a number of *h-votes* and *l-votes*. The number of h-votes and l-votes indicates the probability that $s_{opt} \leq s$ and $s_{opt} > s$, respectively. Assuming that P is a multiple of r (we will discuss later for the cases where this assumption is not true), the PI pilot simulation is described by the following steps:

PI Pilot Simulation

1. For all $s \in I_r$, perform the following steps concurrently:

   (a) Start P/r independent runs concurrently, one processor for each run, to simulate the target system starting from state s for a length T_P. For each run, if the resultant estimate $\hat{\mu}(s, T_P) \geq \hat{\mu}(s, T'_P)$ for some pre-determined intermediate time point T'_P, $0 \leq T'_P < T_P$, we say state s 'wins' an *h-vote*, otherwise (i.e., $\hat{\mu}(s, T_P) < \hat{\mu}(s, T'_P)$) s wins an *l-vote*.

   (b) Calculate the total numbers of h-votes and l-votes won by state s. If the number of h-votes is greater than l-votes, label s with an 'h', otherwise label s with an 'l'.

2. Compute an interval $[a, b]$ based on the following rules:

   (a) If all states are labeled with $l's$, let $a = b = 1$.

   (b) If all states are labeled with $h's$, let $a = b = r$.

   (c) Otherwise, let $a = \max\{k, 1\}$ such that state $k + 1$ is the only state labeled l in $\{1, 2, \ldots k + 1\}$. Let $b = \min\{k, r\}$ such that state $k - 1$ is the only state labeled h in $\{k-1, k, \ldots, r\}$.

3. For each processor $i, 1 \leq i \leq P$, randomly select an initial state from $[a, b]$.

In the actual production runs, the processors concurrently simulate the system for a fixed length $T > 0$ using the initial states obtained from the PI pilot runs and then compute an estimate for μ using estimator $\hat{\mu}_1(P, T)$. We refer to this method as the MR-PI approach.

In the PI pilot simulation, step 1 performs a *test* for each state s, in which P/r independent pilot runs are performed concurrently. For each run, if $\hat{\mu}(s, T_P) \geq \hat{\mu}(s, T'_P)$ (assuming $T'_P \geq T_0$), it gives a strong indication that $\hat{\mu}(s, T)$ is converging toward μ from below. Hence, we 'guess' that $s \leq s_{opt}$ by issuing an h-vote. Similarly, if $\hat{\mu}(s, T_P) < \hat{\mu}(s, T_0)$, we guess $s > s_{opt}$ by issuing an l-vote. The outcome of the test (i.e., $s_{opt} \geq s$ or $s_{opt} < s$) is determined by the majority of the votes (we break a tie by favoring l-votes). Based on these tests, step 2 obtains an interval $[a, b]$ that is likely to contain s_{opt}. The lower bound a is the largest number such that all states in $I_a = \{1, 2, \ldots, a\}$ are labeled with an h (so that we are 'confident' in predicting that $s_{opt} \geq a$). If $I_a = \emptyset$ (i.e., state 1 is labeled with l), we let $a = 1$. The upper bound b is determined in the same manner.

Let $p(s, T_P)$ denote the probability of each guess (as a result of a pilot run) being correct for state s. Apparently, $p(s, T_P)$ increases as T_P increases. Also, $p(s, T_P)$ is a function of the initial state s because the convergence rate typically depends on the initial state. This is illustrated in Figure 1, where the top curve (corresponding to s_2) shows a stronger convergence tendency in the initial transient period. Thus, we expect that $p(s_2, T_P) > p(s_1, T_P)$.

Let $p_h(s, T_P)$ and $p_l(s, T_P)$ represent the probabilities that state s is labeled h and l,

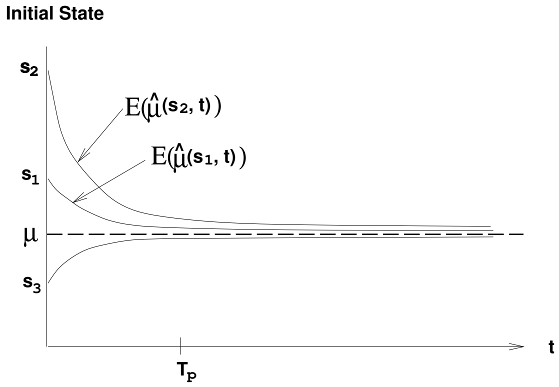

Figure 1: Example convergence curves of $\hat{\mu}(s,t)$.

respectively. Then

$$p_h(s,T_P) = \begin{cases} \sum_{i=\lfloor \frac{P}{2r} \rfloor +1}^{P/r} \begin{pmatrix} \frac{P}{r} \\ i \end{pmatrix} (p(s,T_P))^i (1-p(s,T_P))^{\frac{P}{r}-i} & \text{if } 1 \leq s \leq s_{opt} \\ \sum_{i=\lfloor \frac{P}{2r} \rfloor +1}^{P/r} \begin{pmatrix} \frac{P}{r} \\ i \end{pmatrix} (1-p(s,T_P))^i (p(s,T_P))^{\frac{P}{r}-i} & \text{if } r \geq s > s_{opt} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$p_l(s,T_P) = \begin{cases} \sum_{i=\lceil \frac{P}{2r} \rceil}^{P/r} \begin{pmatrix} \frac{P}{r} \\ i \end{pmatrix} (p(s,T_P))^i (1-p(s,T_P))^{\frac{P}{r}-i} & \text{if } r \geq s > s_{opt} \\ \sum_{i=\lceil \frac{P}{2r} \rceil}^{P/r} \begin{pmatrix} \frac{P}{r} \\ i \end{pmatrix} (1-p(s,T_P))^i (p(s,T_P))^{\frac{P}{r}-i} & \text{if } 1 \leq s \leq s_{opt} \\ 0 & \text{otherwise.} \end{cases}$$

Then the probability that the PI pilot simulation finds an interval containing s_{opt} is given by:

$$\begin{aligned} prob\{a \leq s_{opt} \leq b\} &= 1 - (prob\{a > s_{opt}\} + prob\{b < s_{opt}\}) \\ &= 1 - \left(\Pi_{s=1}^{s_{opt}+1} p_h(s,T_P) + \Pi_{s=s_{opt}-1}^{r} p_l(s,T_P) \right). \end{aligned}$$

Assume that $p(s,T_P) > 0.5$ for any s and T_P. If we assign the outcome of each vote a numerical value (e.g., 1 for h-vote, 0 for l-vote), then based on the Law of Large Numbers

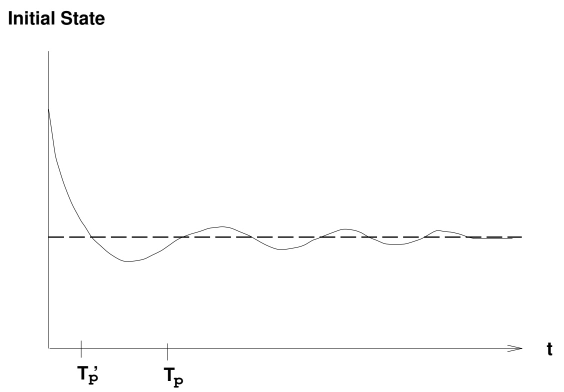

Figure 2: An example of a convergence curve.

we can see that $prob\{a \leq s_{opt} \leq b\} \to 1$ as $P \to \infty$. That is, the region $[a, b]$ will almost always contain μ when $P \to \infty$. Also, the interval of $[a, b]$ is expected to decrease as P increases.

Let $w_{a,b} = b - a + 1$ denote the 'width' of the interval $[a, b]$. It can be derived that

$$E(w_{a,b}) = r - \sum_{s=1}^{r}(s-1)[p_l'(s+1, T_P)\Pi_{i=1}^{s} p_h(s, T_P)] - \sum_{s=1}^{r}(r-s)[p_h'(s-1, T_P)\Pi_{i=s}^{r} p_l(s, T_P)],$$

where

$$p_l'(s+1, T_P) = \begin{cases} p_l(s+1, T_P) & \text{if } s+1 \leq r \\ 1 & \text{otherwise,} \end{cases}$$

and

$$p_h'(s-1, T_P) = \begin{cases} p_h(s-1, T_P) & \text{if } s-1 \geq 1, \\ 1 & \text{otherwise.} \end{cases}$$

We have assumed that $\hat{\mu}(s, T)$ converges monotonically toward μ. It should be noted that this is not a necessary condition to apply the PI approach because the purpose of the PI pilot simulation is simply to obtain an initial state that is relatively close to μ. For example, for a convergence curve as shown in Figure 2 (where the convergence is not monotonic), the PI technique is viable when T_P' and T_P are properly chosen.

				a	b						
State:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
h–vote:	**100**	**100**	**100**	**100**	**43**	**0**	**0**	**0**	**0**	**0**	**0**
l–vote:	**0**	**0**	**0**	**0**	**57**	**100**	**100**	**100**	**100**	**100**	**100**
Label:	**h**	**h**	**h**	**h**	**l**	**l**	**l**	**l**	**l**	**l**	**l**

Figure 3: An example of the PI pilot simulation. The queue length process of an M/M/1/10 queue is considered, where $P = 1000$, $T_P = 20$, $\lambda = 90$, and $\mu = 100$. The pilot simulation obtains an interval [3,4] for s_{opt}.

An application of PI pilot simulation is shown in Figure 3 for an M/M/1/10 queue. The state of the queue is represented by the queue length. Let λ and μ represent the average arrival rate and service rate, respectively of the queue. For $\lambda = 90, \mu = 100, P = 1000$, and $T_p = 20$, the PI pilot simulation obtains interval [3,4] for the initial queue length (the analytical result for the average queue length is 3.97). Note that for the PI approach, if $P < r$, we may simply group I_r into P subsets and apply the PI pilot simulation by treating each aggregated state as a single state. Then an initial state may be randomly chosen from the original states corresponding to the resultant aggregated state. Some experimental results of the PI multiple replication approach are given in Section 6. Before presenting these experimental results we consider another approach (i.e., parallel regenerative simulation) in the next section.

4 Regenerative Simulation

In the multiple replication simulation, the target simulation model is independently simulated for a number of times, resulting in a set of IID *trajectories* (i.e., sample paths) of the system. A point estimate and an interval for μ are constructed based on a set of IID samples resulting from the trajectories. In the regenerative simulation, on the other hand, only one trajectory is computed. The trajectory is partitioned at some time points into a number of IID subtrajectories, from which a set of IID samples are obtained to allow construction of a point estimate and an interval for μ.

In this section, we first review the regenerative methodology (Sections 4.1 and 4.2). Then we propose two techniques (Sections 5 and 5.1) that enhance the applicability of the regenerative approach for parallel simulation.

4.1 Regenerative Simulation Primer

A random process is said to be *regenerative* if there exists a state $\mathbf{r}$ such that the portions of the process defined in the time intervals $[\beta_1, \beta_2), [\beta_2, \beta_3), \ldots$ are IID replications, where β_n is the n^{th} time point when the system enters state $\mathbf{r}$. The state $\mathbf{r}$ is referred to as a *regeneration state*. The time interval between two consecutive *regeneration times* (i.e., the entrance time points to state $\mathbf{r}$) is called a *regeneration cycle*. For example, in a GI/G/1 queue, the system regenerates whenever a job arrives at an empty queue (Figure 4).

Define $\omega_n = \beta_{n+1} - \beta_n$, and

$$Y_n = \int_{\beta_n}^{\beta_n+1} X(t)dt.$$

That is, ω_n is the *length* of the n^{th} regeneration cycle and Y_n is the n^{th} observation computed by the processor assigned to the simulation time interval $[\beta_n, \beta_{n+1})$. Then

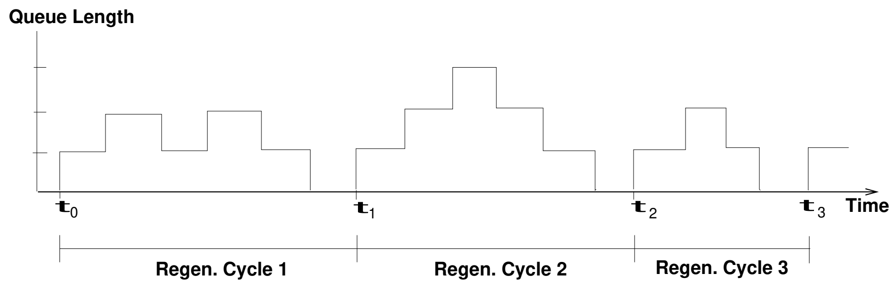

Figure 4: A GI/G/1 queue trajectory. The system regenerates (i.e., a job arrives at an empty queue) at t_0, t_1, t_2, and t_3.

$\{(Y_n, \omega_n), n \geq 1\}$ are IID random vectors and $\mu = E(Y_n)/E(\omega_n)$ [3]. Let $\aleph$ be the total number of regeneration cycles simulated. Then it can be shown that the following estimator for μ, denoted $\hat{\mu}_2(\aleph)$, is *strongly consistent* for μ as $\aleph$ increases (i.e., $\hat{\mu}_2(\aleph) \rightarrow \mu$ with probability one as $\aleph \rightarrow \infty$) [3]:

$$\hat{\mu}_2(\aleph) = \frac{\sum_{i=1}^{\aleph} Y_i}{\sum_{i=1}^{\aleph} \omega_i}.$$

To construct a confidence interval for $\hat{\mu}_2(\aleph)$, two well-known methods are the *classical* approach [3] and the *jackknife* approach [16]. We briefly describe the classical approach below.

Let $Z_i = Y_i - \mu\omega_i$. Then Z_i*'s* are IID random variables with mean 0. Let σ_Z^2 denote the variance of Z_i. By the central limit theorem,

$$\frac{\bar{Z}(\aleph)}{\sqrt{\sigma_Z^2/\aleph}} \Rightarrow \mathbf{N(0,1)} \text{ as } \aleph \rightarrow \infty, \tag{1}$$

where '$\Rightarrow$' denotes convergence in distribution. The sample variance of Z_i, denoted $\hat{\sigma}_Z^2(\aleph)$, is given by:

$$\hat{\sigma}_Z^2(\aleph) = \hat{\sigma}_Y^2(\aleph) + (\hat{\mu}_2(\aleph))^2\hat{\sigma}_\omega^2(\aleph) - 2\hat{\mu}_2(\aleph)\hat{Cov}(Y_i, \omega_i),$$

where $\hat{\sigma}_Y^2(\aleph) = \sum_{i=1}^{\aleph}(Y_i - \bar{Y}(\aleph))^2/(\aleph - 1)$, $\hat{\sigma}_\omega^2(\aleph) = \sum_{i=1}^{\aleph}(\omega_i - \bar{\omega}(\aleph))^2/(\aleph - 1)$, and

$\hat{Cov}(Y_i, \omega_i) = \sum_{i=1}^{\aleph}(Y_i - \bar{Y}(\aleph))(\omega_i - \bar{\omega}(\aleph))/(\aleph - 1)$ are respectively, the sample variance of Y_i, the sample variance of ω_i, and the sample covariance of Y_i and ω_i. It can be shown that $\hat{\sigma}_Z^2(\aleph) \to \sigma_Z^2$ as $\aleph \to \infty$. Because

$$\frac{\bar{Z}(\aleph)}{\bar{\omega}(\aleph)} = \frac{\bar{Y}(\aleph)}{\bar{\omega}(\aleph)} - \mu = \hat{\mu}_2(\aleph) - \mu,$$

from (1), we have:

$$\frac{\hat{\mu}_2(\aleph) - \mu}{\sqrt{\hat{\sigma}_Z^2(\aleph)/\aleph(\bar{\omega}(\aleph))^2}} \Rightarrow \mathbf{N(0,1)} \text{ as } \aleph \to \infty, \tag{2}$$

Then, an approximate 100(1-α) percent confidence interval for μ is given by:

$$\hat{\mu}(\aleph) \pm \frac{z_{1-\alpha/2}\sqrt{\hat{\sigma}_Z^2(\aleph)/\aleph}}{\bar{\omega}(\aleph)}.$$

4.2 Parallel Regenerative Simulation

A key property for a regenerative system is that at a regeneration time point the system becomes totally independent of any previous states. This property provides potential parallelism for simulation because each processor can compute one or more regeneration cycles independently. This section reviews an estimator for μ based on a parallel regenerative approach proposed by Glynn and Heidelberger.

To illustrate parallel regenerative simulation, consider the GI/G/1 trajectory illustrated in Figure 4. The trajectory in the time interval $[t_0, t_3)$ can be partitioned into three subtrajectories at t_0, t_1 and t_2, respectively, and each can be computed by a processor independently. Therefore, this approach can be treated as a special case of *time-parallel* simulation (e.g., [2, 9, 18, 19]) in which parallelism is obtained through partitioning the time domain of the simulation model.

Let τ_r be the *real* (i.e., wall clock) simulation execution time for a simulation time length τ. Assume that the real execution time is linear to the simulation time length.

That is, there exists a constant $\eta > 0$, such that $\tau_r = \eta\tau$. Suppose τ is the *desired* simulation time length. For balancing the workload it is desirable that each processor simulates τ/P time units (or equivalently τ_r/P real time units). Unfortunately, because a regenerative simulation requires that an integer number of regeneration cycles be simulated, and because the regeneration cycle length is a random variable, the simulation run length of each processor is also a random variable. Therefore, the perfect workload partitioning (i.e., all processors execute exactly τ/P simulation time units) can not be achieved.

In parallel regenerative simulation, a key problem is how to stop the processors. There exists a number of 'stopping rules', each resulting in a different estimator for μ. An example of a stopping rule is the *static computation assignment* [1], in which each processor completes a fixed number of regeneration cycles and then stops. Estimators based on a variety of stopping rules have been studied by Glynn and Heidelberger [7, 8, 11]. They showed that some intuitive stopping rules could cause severely biased results. Glynn and Heidelberger [8, 11] have proposed a strongly consistent estimator [8, 11] that uses τ/P as a *guideline* (because processor execution time is a random variable) stopping time. This estimator forms the basis of our PR approach which will be discussed later. For consistency with our notations, this estimator is denoted by $\hat{\mu}_3(P, \frac{\tau}{P})$.

For $\hat{\mu}_3(P, \frac{\tau}{P})$, all processors start simultaneously to simulate the system from a fixed regeneration state. For each processor, when the *system enters the regeneration state and at least τ/P time units have been elapsed*, the processor stops. Let $N_i(\tau/P)$ denote the number of regeneration cycles completed on processor i in τ/P time units. Then processor i will simulate exactly $N_i(\tau/P) + 1$ regeneration cycles. Let Y_{ij} and ω_{ij} denote the j^{th} observation and the j^{th} regeneration cycle length (RCL), respectively, on processor

i. Then $\hat{\mu}_3(P, \frac{\tau}{P})$ is obtained by:

$$\hat{\mu}_3(P, \frac{\tau}{P}) = \frac{\sum_{i=1}^{P} \sum_{j=1}^{N_i(\tau/P)+1} Y_{ij}}{\sum_{i=1}^{P} \sum_{j=1}^{N_i(\tau/P)+1} \omega_{ij}}. \tag{3}$$

It can be shown that $\hat{\mu}_3(P, \tau/P)$ is strongly consistent for μ as P increases [11]. Because $\sum_{j=1}^{N_i(\tau/P)+1}(Y_{ij} - \mu\omega_{ij})$, for $1 \leq i \leq P$, are IID random numbers, based on the central limit theorem, as $P \to \infty$,

$$\frac{E(\omega_{ij})E(N_i(\tau/P)+1)(\hat{\mu}_3(P, \frac{\tau}{P}) - \mu)}{\sigma(\tau/P)/\sqrt{P}} \Rightarrow \mathbf{N}(\mathbf{0}, \mathbf{1}), \tag{4}$$

where $\sigma^2(\tau/P)$ is the variance of $\sum_{j=1}^{N_i(\tau/P)+1}(Y_{ij} - \mu\omega_{ij})$. Therefore, an approximate $100(1-\alpha)$ percent confidence interval can be obtained by:

$$\hat{\mu}_3(P, \frac{\tau}{P}) \pm z_{1-\alpha/2} \frac{\hat{\sigma}(\tau/P)}{\sum_{i=1}^{P} \sum_{j=1}^{N_i(\tau/P)+1} \omega_{ij}/\sqrt{P}},$$

where $\hat{\sigma}^2(\tau/P)$ is the sample variance of $\sum_{j=1}^{N_i(\tau/P)+1}(Y_{ij} - \mu\omega_{ij})$. Let $T(P)$ denote the completion time of the regenerative simulation using P processors. Also, let γ_i denote the residual time at τ/P for processor i. That is,

$$\gamma_i = \left(\sum_{j=1}^{N_i(\tau/P)+1} \omega_{ij} \right) - \frac{\tau}{P}.$$

Then

$$T(P) = \max_{1 \leq i \leq P} \{ \sum_{j=1}^{N_i(\tau/P)+1} \omega_{ij} \} = \tau/P + \max_{1 \leq i \leq P} \{\gamma_i\}.$$

It can be shown that [5, Section 4.2]

$$\tau/P + \mu_\gamma(\tau/P) \leq E(T(P)) \leq \tau/P + \mu_\gamma(\tau/P) + \left(\frac{\sigma_\gamma(\tau/P)(P-1)}{\sqrt{2P-1}} \right), \tag{5}$$

where $\mu_\gamma(\tau/P)$ and $\sigma_\gamma(\tau/P)$ are the mean and the variance of γ_i, respectively.

5 The Selection of Regeneration State

For a regenerative process, there usually exists multiple regeneration states. For example, all states in a homogeneous Markov chain are regenerative. Therefore, to apply the parallel regenerative simulation, one concern is how to choose a regeneration state. In this section, we first argue that it is advantageous to use regeneration states that have high occurrence frequencies. Then we present a pilot simulation that selects regeneration states which occur frequently.

For the parallel regenerative approach, we define the degree of parallelism to be the expected number of regeneration cycles in the simulation time interval $[0, \tau]$. That is, the degree of parallelism denotes the expected number of processors that we can use so that each processor can finish at least one regeneration cycle for a total simulation length τ. Apparently, to obtain a high degree of parallelism, it is desirable that we use a regeneration state that occurs relatively frequently.

There are two other potential advantages using a frequently occurring regeneration state (FORS). Consider $\hat{\mu}_3(P, \frac{\tau}{P})$, when regeneration occurs frequently, after the guideline stopping time has passed, a processor may quickly reach a regeneration point and terminates. Therefore, a short simulation completion time results.

Consider a process which has $r > 1$ regeneration states. Let $\gamma_{i,k}$ denote the residual time at τ/P for processor i when using regeneration state k, $1 \leq k \leq r$. Assume that the RCL corresponding to each regeneration state k, denoted Ω_k, is exponentially distributed with a mean $1/\lambda_k$. Due to the memoryless property of an exponential distribution, $\gamma_{i,k}$ and Ω_k have the same distribution. Therefore, for regeneration state k, the simulation completion time, denoted $T_k(P)$, is the P^{th} order statistic of the exponential distribution with mean $1/\lambda_k$. It can then be shown that

$$E(T_k(P)) = P\int_{t=0}^{\infty} te^{-\lambda_k t}(1-e^{-\lambda_k t})^{P-1}dt = \frac{1}{\lambda_k}\sum_{i=1}^{P}\frac{1}{i}.$$

Therefore, in this case, the expected simulation completion time strictly decreases as we use a regeneration state that has a smaller expected RCL. This relationship is also illustrated in Figures 5 and 6 by the queue length process of a loss M/M/1/10 queue where $\lambda/\mu = 0.7$. Note that the RCL of this process is not exponentially distributed.

Let $\overline{T_k(P)}$ and $\underline{T_k(P)}$, respectively, denote the upper and the lower bound on $T_k(P)$ given by (5). Then as a direct result of (5), we can see that if $E(\Omega_k) < E(\Omega'_k) \Longrightarrow \sigma_{\gamma_k} < \sigma_{\gamma_{k'}}$, where '$\Longrightarrow$' represents 'imply' and σ_{γ_k} denotes the variance of $\gamma_{i,k}$, the following condition holds:

$$E(\Omega_k) < E(\Omega'_k) \Longrightarrow (\underline{T_k(P)} < \underline{T'_k(P)}) \wedge (\overline{T_k(P)} < \overline{T'_k(P)}).$$

The third potential advantage of using the FORS regeneration state is that a FORS results in a large number of observations in the simulation interval. This, generally, is helpful in reducing the variance of the estimate. We have given some potential advantages of using a FORS for the parallel regenerative simulation. When *a priori* knowledge of a FORS is unavailable, we propose using the following pilot simulation:

FORS Pilot Simulation

1. For all i, $1 \leq i \leq P$, processor i randomly chooses an initial state and simulates the system for some short time period. Record the occurrences of each regeneration state.

2. For each processor i, identify the most frequently occurring regeneration state, denoted by $\mathbf{r}_i$, in the step 1 simulation run.

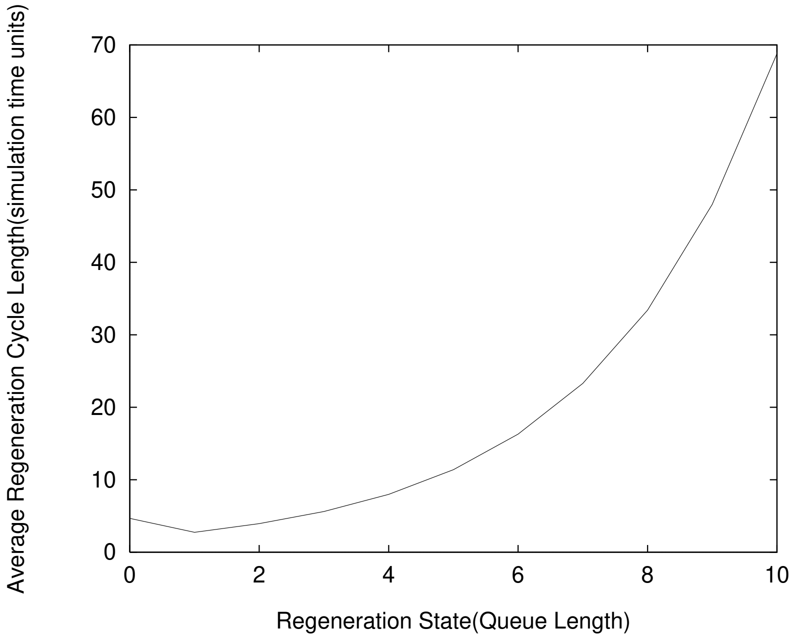

Figure 5: Average RCL for the M/M/1/10 queue, where $\lambda = 70$, $\mu = 100$, and $P = 100$. The guideline stopping time is 100.

3. Select the state that appears the most times in $\{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_P\}$. Break a tie arbitrarily.

Note that the heuristic pilot simulation presented above is not guaranteed to select the most frequently occurring state. The accuracy of the pilot simulation generally increases as the pilot simulation length increases. However, because the purpose of the pilot simulation is simply to obtain a regeneration state that occurs relatively frequently, we may determine the pilot simulation length somewhat arbitrarily. For the M/M/1/10 queue where $\lambda/\mu = 0.7$, our experiments show that the FORS pilot simulation almost always successfully selects the true most frequently occurring state even when each processor simulates as few as 20 job arrivals when $P \geq 100$.

The regenerative structure of a random process provides statistical convenience for constructing an estimate for μ, and allows us to partition the simulation easily for multi-

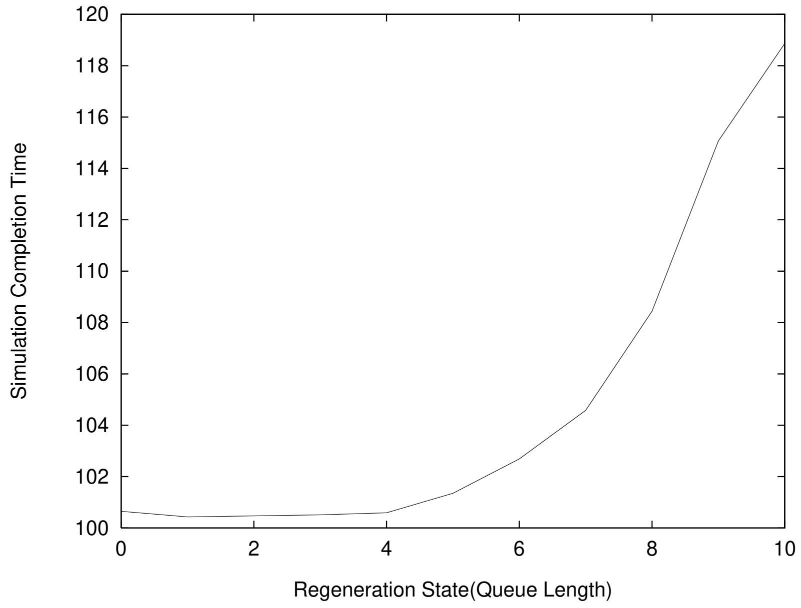

Figure 6: Parallel regenerative simulation completion time for the M/M/1/10 queue, where $\lambda = 70$, $\mu = 100$, and $P = 100$. The guideline stopping time is 100.

ple processor execution. However, the regenerative approach requires that the expected RCL be relatively short so that sufficient IID observations can be obtained in the desired simulation interval. In practice, many systems of interest (e.g., a congested network) do not have a short (or even finite) expected RCL. In such cases, in order to obtain enough observations, we may use approximate regeneration cycles by relaxing the regeneration condition. The concept of approximate regeneration was originally suggested by Crane and Iglehart [4]. In the next section, we present an approximate regeneration technique using substate matching.

5.1 Parallel Regenerative Simulation with Substate Matching

Let $V = \{v_1, v_2, \ldots, v_m\}$ be the set of state variables of the target simulation model and $v_k(t)$, $1 \leq k \leq m$, denotes the value of v_k at simulation time t. A state variable $v_k \subset V$ is *redundant* if the v_k can be uniquely determined by any other state variables in V.

Assume that no state variables in V are redundant. Then we say that process $\mathbf{X}$ can be sufficiently described by the process $\mathbf{V}=\{v_1(t), v_2(t), \ldots, v_m(t), t \geq 0\}$ if there exists a set $W = \{v_{k_1}, v_{k_2}, \ldots, v_{k_{m'}}\} \subseteq V$, $1 \leq m' \leq m$, and a function f such that $X(t) = f(W(t))$ for all $t \geq 0$, where $W(t) = (v_{k_1}(t), v_{k_2}(t), \ldots, v_{k_{m'}}(t))$. If there does not exist any set $W' \subset W$ and a function f' such that $X(t) = f'(W'(t))$ for all $t \geq 0$ is satisfied then we refer to set W as an *essential* set for $\mathbf{X}$. That is, the state of $\mathbf{X}$ directly depends on the state variables in an essential set.

Assume that process $\mathbf{V}$ is regenerative. This is not a restrictive assumption because Crane and Iglehart [3] have shown that with appropriate arrangement of the state vector, a simulation model can always be structured as a regenerative process. If process $\mathbf{V}$ enters regeneration state $\mathbf{r} = (r_1, r_2, \ldots, r_m)$ at time t (i.e., $r_k = v_k(t)$ for all $1 \leq k \leq m$), we say that the process regenerates on $\mathbf{r}$ at t.

Define *substate* $\mathbf{r}_W$ to be the portion of state $\mathbf{r}$ corresponding to set W. For example, if $W = \{v_1, v_2, v_4\}$, $\mathbf{r}_W=\{r_1, r_2, r_4\}$. We say that the system *partially regenerates*[1] on $\mathbf{r}_W$ at time t if $v_{k_j}(t) = r_{k_j}$ for all $1 \leq j \leq m'$. A partial regeneration cycle length (PRCL) is the time interval between two consecutive occurrences of partial regeneration. Then we can see that for any regeneration state $\mathbf{r}$, the expected RCL on $\mathbf{r}$ is no larger than the expected PRCL $\mathbf{r}_W$. This is obvious because if the system partially regenerates on $\mathbf{r}$ at any time t, the system must also regenerate on $\mathbf{r}_W$ at t, and the converse is not true. Therefore, if the expected PRCL corresponding $\mathbf{r}_W$ is finite and when $t \rightarrow \infty$, it is almost certain that number of partial regeneration cycles (corresponding to $\mathbf{r}_W$) will be larger than the original regeneration cycles (corresponding to $\mathbf{r}$). Based on this property, we consider the following approximate regenerative approach:

[1]The term 'partial regeneration' was introduced in [15].

Partially Regenerative (PR) Simulation

1. Identify an essential set W and a substate $\mathbf{r}_W$ such that the target system partially regenerates on $\mathbf{r}_W$ relatively frequently.

2. Initialize the state variables corresponding to W with substate $\mathbf{r}_W$.

3. Initialize the state variables corresponding to $W' = V - W$ (where '-' is the set difference operator) appropriately.

4. For all i, $1 \leq i \leq P$, processor i independently simulates the target system until both of the following conditions are satisfied:

   (a) The system partially regenerates on $\mathbf{r}_W$.

   (b) At least τ/P simulation time units have been simulated.

5. Let $N_i'(\tau/P)$ denote the number of partial regeneration cycles completed on processor i in τ/P simulation time units and let Y_{ij}' and ω_{ij}' denote the j^{th} observation and the j^{th} PRCL, respectively, on processor i. Compute an estimate for μ as follows:

$$\hat{\mu}_4(P, \frac{\tau}{P}) = \frac{\sum_{i=1}^{P} \sum_{j=1}^{N_i'(\tau/P)+1} Y_{ij}'}{\sum_{i=1}^{P} \sum_{j=1}^{N_i'(\tau/P)+1} \omega_{ij}'}.$$

The PR approach obtains an estimate for μ using (3) by treating the partial regeneration cycles (i.e., on $\mathbf{r}_W$) as though they are true regeneration cycles. We refer $\mathbf{r}_W$ as the *matching substate* because a partial regeneration cycle is determined by a match between the system substate corresponding to W and $\mathbf{r}_W$. Due to the relaxation of the regeneration condition, the partial regeneration cycles are unlikely to be IID. Consequently, the simulation results could be biased. Nevertheless, by applying Gunther's almost renewal process theorems [10], it can be shown that $\hat{\mu}_4(P, \frac{\tau}{P})$ is a strongly consistent estimator for

μ as $\tau \to \infty$. When strong correlation is expected among the partial regeneration cycles, we may apply the *batch means* method by collecting only one observation for every $b > 1$ consecutive partial regeneration cycles. When b is large, under some mild conditions, it can be shown that the resultant observations are approximately uncorrelated [14].

Application

To illustrate the PR approach, we consider an example shown in Figure 7 which represents a packet-switched local area network (LAN) of four hosts (i.e., nodes 1 to 4) and a gateway (i.e., node 5) that connects the LAN to other networks. Each node is modeled by a loss G/G/1/K queue, where arriving packets that find the queue full are dropped, representing a packet loss due to a buffer overflow. For each departing packet from a host, the next node to enter is chosen randomly from those connected to the host.

Let QL_i denote the queue length of node $i, 1 \leq i \leq 5$. Suppose we are interested in the average congestion level (i.e., the queue length) of the gateway. Then the essential set $W = \{QL_5\}$. For the matching substate $\mathbf{r}_W$, observe that for any G/G/1/K queue where $K < \infty$, the queue length being zero (respectively, K) will always have a relatively high occurrence frequency in the long run when λ/μ is sufficiently smaller (larger) than 1. Therefore, we may simply let $\mathbf{r}_W$=K and $\mathbf{r}_W$=0 for $\lambda/\mu > 1$ and $\lambda/\mu < 1$, respectively. Otherwise, the FORS pilot simulation can be applied to estimate a frequently occurring substate for $\mathbf{r}_W$. Finally, the queues corresponding to nodes 1 to 4 have to be initialized appropriately to avoid large initial transient. This can be achieved by using the PI pilot simulation discussed in Section 3.1.

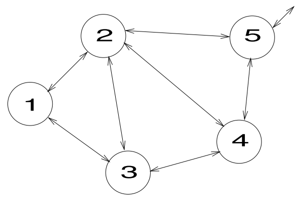

Figure 7: An open queueing network of five nodes. Each node contains a queue and a server to retain and serve the arriving jobs.

6 Experimental Results

This section empirically evaluates the performance of the MR-PI and the PR approaches on some queueing models.

6.1 Loss G/G/1/K Queues

We first consider a loss G/G/1/K queue which is often used in modeling multiplexers or other components in communication networks. The loss G/G/1/K model can be represented by a regenerative process $\{NAT(t), QL(t), ST_j(t), t \geq 0\}$, where NAT, QL, and $ST_j, 1 \leq j \leq K$ are the time interval before the next arrival time, queue length, and the remaining service time of the j^{th} job in the buffer ($ST_j = 0$ for $j > QL$), respectively. Note that any process that contains only a proper subset of these three state variables is not a regenerative process for general interarrival and service times. Supposed we are interested in the average queue length. Then for the PR approach, essential set $W = \{QL\}$. For the PR and the PI runs, the initial value of NAT is drawn from the interarrival time distribution, while the initial values of ST_i, $1 \leq j \leq QL$, are drawn independently from the service time distribution. For the MR-D runs, the queue are idle initially.

The M/M/1/100 and $\Gamma(2)$/D/1/100 queues are used, where $\Gamma(2)$ denotes the gamma distribution with a shape parameter 2. Note that the M/M/1/K model is considered because an analytical solution is readily available. In our experiments, we let $\mu = 100$ and $\lambda = 50, 70, 90, 100$ and 110, respectively to represent low to overloaded traffic conditions. Under each traffic condition, $P = 10^2, 10^3$, and 10^4, respectively. For the PR simulation, the desired simulation length is $\tau = 10^4$ simulation time units. Thus, the guideline stopping time is $10^4/P$ simulation time units. The matching substate $\mathbf{r}_W$ in each case is obtained by applying the FORS pilot simulation. For the MR-D simulation, a variety of warm-up period lengths ranged from 0% to 90% of the total simulation length are tested.

For a fair comparison, in each case all three approaches use the same overall simulation time (including the pilot simulations). That is, the simulation lengths for the MR-D and the MR-PI runs are given by

$$\tau_{MR-D}(P) = \frac{\sum_{i=1}^{P} \sum_{j=1}^{N_i'(\tau_r/P)+1} \omega_{ij}' + \Delta\tau_{PR}}{P},$$

and

$$\tau_{PI}(P) = \tau_{MR-D}(P) - \Delta\tau_{PI},$$

respectively, where τ_{PI} and τ_{PR} denote the pilot simulation run length for the MR-PI and the PR simulations. In our experiments, τ_{PI} and τ_{PR} are both equal to 20% of the guideline stopping time of the PR runs (i.e., $2000/P$ simulation time units).

Experimental results are given in Tables 1 and 2, where MR-D$_x$ represents MR-D simulation with the first x% of the simulation time interval deleted. We define the *statistical efficiency* of a simulation to be the ratio of the simulation error to the total simulation time spent. The normalized simulation errors shown in the square brackets in the tables thus indicate the statistical efficiencies of the simulation runs.

The results show that the PR approach produces highly accurate estimates for all tested cases. Typical errors against the expected values are well under 1.0%. The MR-D

Table 1: The average queue lengths with approximate 90% confidence intervals of the M/M/1/100 model. Values in row 'EXPECT' are computed analytically. The values in the square brackets show the normalized errors against the expected values. Absence of a value in square brackets denotes an error no more than 1.0%.

Average Queue Length (M/M/1/100)

P	Approach	λ/μ =0.5	λ/μ =0.7	λ/μ =0.9	λ/μ =1.0	λ/μ =1.1
100	EXPECT	1.0	2.333	8.997	50.0	90.007
	PR	1.00(±.01)	2.34(±.03)	9.03(±.32)	48.87(±2.92) [2.3%]	90.01(±.43)
	MR-PI	1.00(±.01)	2.34(±.03)	8.90(±.32)	49.49(±1.53) [1.0%]	90.12(±.31)
	MR-D_0	1.01(±.01)	2.35(±.03)	9.20(±.33) [2.2%]	47.13 (±1.67) [5.7%]	86.54(±.40) [3.8%]
	MR-D_{20}	1.00(±.01)	2.34(±.03)	9.15(±.30) [1.7%]	51.85(±2.12) [3.7%]	90.01(±.33)
	MR-D_{50}	1.00(±.01)	2.32(±.03)	8.94(±.39)	50.30(±2.23)	90.06(±.38)
	MR-D_{90}	1.00(±.02)	2.31(±.07)	9.98(±1.1) [10.9%]	48.56(±3.91) [2.9%]	89.51(±.84)
1000	PR	.99(±.01)	2.63(±.03)	8.83(±.31) [1.9%]	49.03(±1.59) [1.9%]	90.05(±.31)
	MR-PI	1.00(±.01)	2.33(±.03)	8.54(±.22) [5.1%]	46.73(±1.06) [6.5%]	90.64(±.25)
	MR-D_0	1.00(±.01)	2.31(±.02)	8.33(±.21) [7.4%]	38.16(±.84) [23.7%]	60.69(±.73) [32.6%]
	MR-D_{20}	1.00(±.01)	2.35(±.03)	8.56(±.23) [4.9%]	42.19(±.99) [15.6%]	68.44(±.82) [43.5%]
	MR-D_{50}	1.00(±.01)	2.34(±.03)	9.31(±.32) [3.4%]	46.31(±1.16) [7.4%]	81.53(±.78) [9.4%]
	MR-D_{90}	1.03(±.01) [3.0%]	2.37(±.06) [1.7%]	9.12(±.38) [1.3%]	48.08(±1.42) [3.6%]	87.04(±.74) [3.3%]
10000	MR-PI	.99(±.01)	2.63(±.03)	8.98(±.25)	49.28(±.98) [1.4%]	89.94(±.27)
	MR-PI	.95(±.01) [4.0%]	2.43(±.02) [4.3%]	6.38(±.06) [29.1%]	55.18(±.4) [10.4%]	93.96(±.06) [4.4%]
	MR-D_0	.96(±.01) [4.0%]	2.08(±.02) [10.7%]	5.62(±.06) [37.5%]	27.40(±.24) [45.2%]	14.14(±.12) [84.3%]
	MR-D_{20}	1.01(±.01)	2.27(±.02) [2.6%]	6.11(±.07) [32.1%]	30.37(±.28) [39.3%]	16.76(±.15) [81.4%]
	MR-D_{50}	.99(±.01)	2.30(±.03) [1.3%]	6.86±.09) [23.7%]	35.3 4(±.36) [28.9%]	20.10(±.24) [77.5%]
	MR-D_{90}	1.00(±.01)	2.46(±.04) [5.6%]	7.44(±.11) [17.3%]	38.96(±.43) [22.1%]	24.71(±.24) [72.5%]

Table 2: The average queue lengths of the $\Gamma(2)/D/1/K$ model with approximate 90% confidence intervals. Each entry in row 'EXPECT' is estimated by a simulation of 10 independent replications for a total of 10^8 arrivals. The values in the square brackets are the normalized errors against the 'EXPECT' values. Absence of a value in square brackets denotes an error less than 1.0%.

Average Queue Length ($\Gamma(2)/D/1/100$)

P	Approach	λ/μ =0.5	λ/μ =0.7	λ/μ =0.9	λ/μ =1.0	λ/μ =1.1
100	EXPECT	.588	1.045	2.836	50.682(±.358)	97.28
	PR	.588(±.001)	1.046(±.004)	2.83(±.04)	50.89(±2.61)	97.31(±.05)
	MR-PI	.589(±.001)	1.048(±.004)	2.84(±.04)	49.19(±2.52) [3.0%]	97.27(±.05)
	MR-D_0	.589(±.001)	1.047(±.004)	2.83(±.04)	45.41(±2.31) [10.4%]	92.53(±.20) [4.9%]
	MR-D_{20}	.588(±.002)	1.048(±.004)	2.86(±.05)	48.60(±2.71) [4.1%]	97.32(±.05)
	MR-D_{50}	.590(±.002)	1.043(±.006)	2.81(±.05)	48.99(±3.3) [3.3%]	97.29(±.06)
	MR-D_{90}	.589(±.001)	1.041(±.013)	2.81(±.14)	48.34(±4.5) [4.6%]	97.13(±.16)
1000	PR	.588(±.001)	1.045(±.004)	2.83(±.04)	50.71(±1.81)	97.27(±.05)
	MR-PI	.588(±.001)	1.03(±.003)	2.84(±.04)	49.53(±1.31) [2.3 %]	97.29(±.04)
	MR-D_0	.587(±.001)	1.046(±.004)	2.81(±.04)	29.12(±.80) [42.5%]	51.37(±.57) [47.2%]
	MR-D_{20}	.588(±.002)	1.046(±.004)	2.80(±.04)	34.40(±.97) [32.1%]	61.55(±.67) [36.7%]
	MR-D_{50}	.590(±.002)	1.044(±.006)	2.83(±.05)	37.76(±1.20) [25.5%]	75.61(±.73) [22.3%]
	MR-D_{90}	.591(±.005)	1.035(±.01) [1.7%]	2.84(±.10)	41.17(±1.36) [18.8%]	89.35(±.64) [8.1%]
10000	PR	.587(±.001)	1.045(±.004)	2.79(±.03) [1.8%]	50.21 (±1.08)	97.29(±.04)
	MR-PI	.582(±.001)	1.027(±.003) [1.7%]	2.45(±.02) [13.6%]	51.63 (±.41) [1.9%]	97.74(±.02)
	MR-D_0	.578(±.001) [1.7%]	1.018(±.003) [2.6%]	2.37 (±.02) [16.2%]	14.09(±.13) [72.2%]	8.66(±.06) [91.1%]
	MR-D_{20}	.589(±.002)	1.053(±.004) [2.6%]	2.54(±.02) [10.2%]	15.67(±.15) [69.0%]	9.86(±.07) [89.9%]
	MR-D_{50}	.588(±.002)	1.049(±.006)	2.68(±.03) [5.3%]	17.60(±.18) [65.3%]	12.01(±.09) [87.5%]
	MR-D_{90}	.590(±.004)	1.086(±.01) [3.9%]	2.76(±.04) [2.5%]	2 0.22(±.24) [60.1%]	17.15(±.11) [82.4%]

simulation is competitive in accuracy only when the traffic load is low (e.g., $\lambda/\mu = 0.5$). This is because $QL = 0$ is a 'good' initial state for a light traffic load. As P or λ/μ increases, the initial transient increasingly impairs the accuracy of the MR-D approach. When $P > 1000$ and $\lambda/\mu > 0.7$, the errors are significant regardless of the deletion period length. Note that in our experiments, the MR-D runs start with an empty queue. If the MR-D runs start with a full queue, the MR-D approach will be more accurate when λ/μ is large. The accuracy deteriorates as λ/μ decreases.

For the MR-PI approach, the accuracy is very good when $P \leq 1000$. By comparing the MR-PI and the MR-D_0 runs (because they are only different in the initial states), it shows that the MR-PI approach almost always produces better results. In many cases, the MR-PI approach is more accurate than the MR-D approach for all deletion lengths (Note that, for the MR-PI simulation, warm-up deletion can also be applied to further reduce the bias). This is because the PI pilot simulation improves the the initial state. Our results show that except when $\lambda/\mu = 1$ (when the steady-state queue length is unstable), the PI pilot simulation can very accurately locates the steady-state queue length even with a very short pilot run length. This is illustrated in Figure 8.

The major disadvantage of the PR approach is that the simulation completion time of each processor is a random variable. The MR-D and MR-PI approaches, on the other hand, have a fixed completion time. Nevertheless, for the PR simulation, except when $\lambda/\mu \to 1$, each processor reaches a partial regeneration point very quickly. Therefore, the completion time of the three approaches are approximately the same. When λ/μ is near one, however, the queue length becomes unstable, some processors may take a long time to reach a partial regeneration point. This is illustrated in Figures 9, where the simulation completion time increases dramatically when $\lambda/\mu \to 1$.

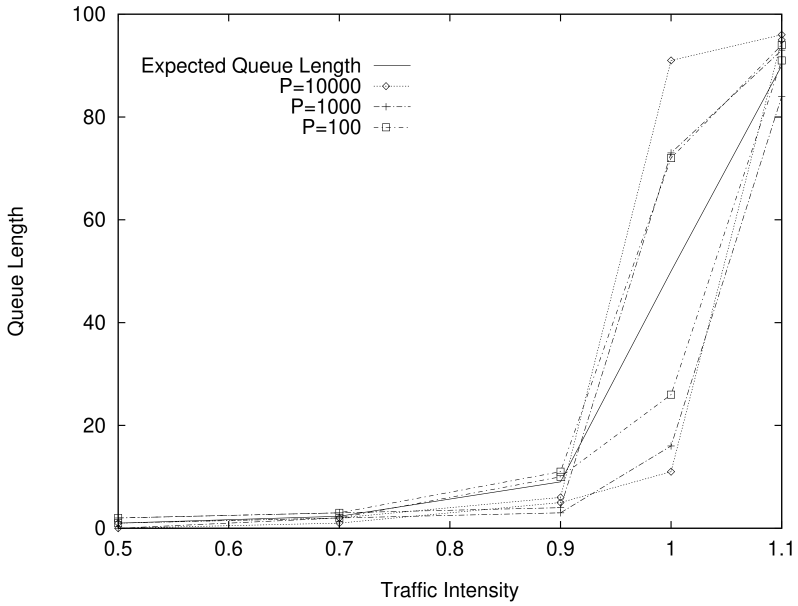

Figure 8: The estimated intervals $[a, b]$ for s_{opt} computed by the MR-PI approach for the M/M/1/100 queue.

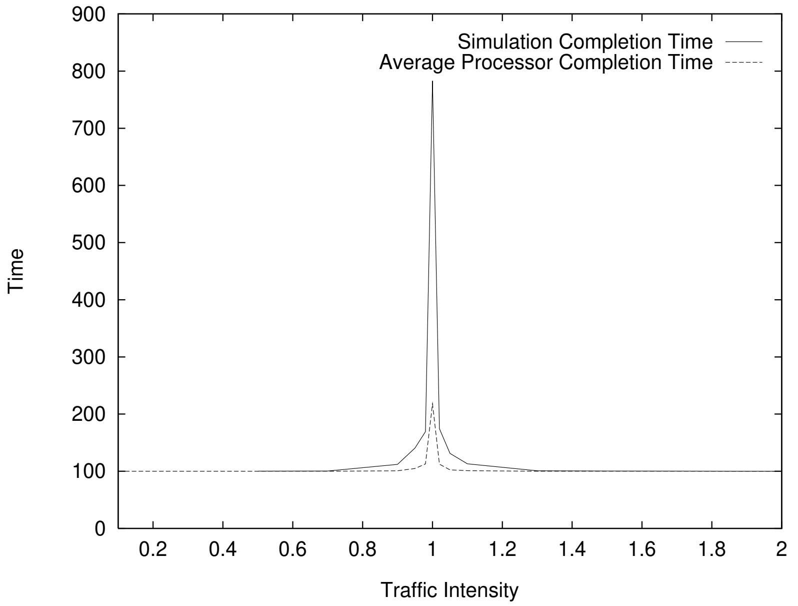

Figure 9: The PR simulation completion time for the M/M/1/100 model for $P = 100$. The guideline stopping time is 100 (simulation time units).

6.2 Closed Queueing Networks

In packet-switched networks, data transmission can be *connection-oriented* or *connectionless*. In the connection-oriented mode, before a transmission between a pair of nodes begins, an end-to-end path called *virtual circuit* (VC) has to be set up. During the connection, all packets are transmitted to the destination node through the virtual circuit. To control the amount of traffic admitted to the VC, an end-to-end flow control mechanism is required. *Sliding-window* [17, Section 5.2.2] is a commonly used flow control mechanism that allows a packet to be transmitted on the VC only if there are less than N_W outstanding packets in the VC from the source node, where N_W is a parameter referred to as the *window size*. When the destination node receives a packet from the source node, the destination node acknowledges the source node upon reception of the packet.

A VC with sliding-window flow control can be modeled by a closed network of G/G/1/K queues [17, Section 5.2.2] as shown in Figure 10. Node 0 is a *virtual* node used to model the arrival process of the source node under the assumption that acknowledgments are transmitted at a higher priority and thus incur no queueing delay. When there are N_W outstanding packets from the source node in the VC, node 0 is empty so that no more packets will be transmitted onto the circuit. Otherwise, packets enter the VC at an arrival rate of λ_0.

Here, we consider a VC of three hops where $N_W = 20$ and $\lambda_0 = \mu_1 = \mu_3 = 1.25\mu_2$. Assume that the average number of packets of, the bottleneck node (i.e., node 2) is of interest. Other parameters in our experiment are: $\lambda_0 = 1$, the desired simulation length $\tau = 10^5$ (i.e., the guideline stopping time is $10^5/P$), and $K = 30$ for all nodes. Also, in our model, the packet lengths follow the $\Gamma(2)$ distribution and each packet is assigned a

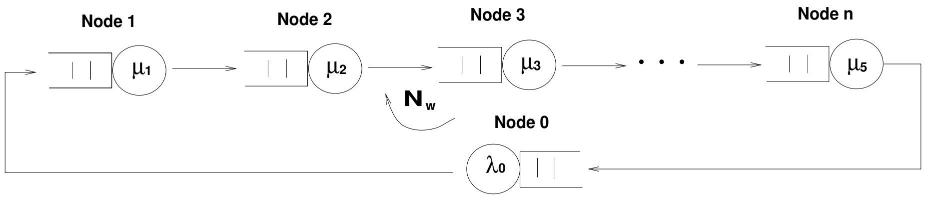

Figure 10: A virtual circuit with a sliding-window control, in which N_W is the window size.

Table 3: The most frequently occurring state of QL_2 (i.e., r_w) and the s_{opt} intervals (i.e., $[a, b]$) computed by the FORS and the PI pilot simulations, respectively, for the VC model.

	$P = 100$	$P = 1000$	$P = 10000$
$\mathbf{r}_W$	16	16	16
$[a, b]$	[14,18]	[15,16]	[15,16]

new length each time when it arrives at a queue.

Compared to the G/G/1/K model in the previous section, the VC model has a much larger state vector and the state variables (e.g., the queue lengths) belonging to different nodes are correlated. To apply the PR simulation, we let $W = \{QL_2\}$. To determine the matching substate $\mathbf{r}_W$, again, we apply the FORS pilot simulation. The lengths for the FORS and the PI pilot runs are 10 simulation time units (i.e., 1% of the guideline stopping time when $P = 100$) for all P's. For both MR-PI and MR-D simulations, each run starts at a state where the packets are evenly distributed among all nodes.

Experimental results of the VC model are shown in Table 4. It shows that the PR simulation produces highly accurate and consistent results (less than 1.0% of error) in all cases. For the MR simulation, the errors are large for $P \geq 1000$. Comparing the MR-PI and the MR-D_0 runs, it appears that the PI pilot simulation significantly reduces the initial transient bias. For the PR simulation, however, the trade-off for the high accuracy

Table 4: The estimated queue lengths of node 2 with approximate 90% confidence intervals. Each entry in column 'EXPECT' is estimated by a simulation of 10 independent replications, each of 10^6 simulation time units in length. The values in the square brackets are the normalized errors against the 'EXPECT' values. Absence of a value in square brackets denotes an error less than 1.0%.

Approach	$P = 100$	$P = 1000$	$P = 10000$
EXPECT	13.177(± .024)	13.177(± .024)	13.177(±.024)
PR	13.29(± 0.15)	13.16(± 0.13)	13.21(± 0.10)
MR-PI	13.26(± 0.15)	13.76(± 0.10)[3.7%]	14.58(± .03)[10.6%]
MR-D_0	12.85(± .14)[2.4%]	11.16(± .12)[15.3%]	7.86(± .04)[40.3%]
MR-D_{20}	13.29(± .16)	12.38(± .14)[6.0%]	8.51(± .05)[35.4%]
MR-D_{50}	13.20(± .23)	13.03(± .15)[1.1%]	9.33(±.06)[29.2%]
MR-D_{90}	13.25(± .35)	13.65(± .19)[3.5%]	11.30(± .08)[14.2%]

is the extra completion time (Figure 11).

7 Concluding Remarks

This paper investigates multiple replication and parallel regenerative approaches for massively parallel simulation. For multiple replication simulation, due to the lack of *a priori* knowledge about the steady-state conditions, an arbitrarily selected initial state is often used. This can result in significant bias in the simulation outcome. A pilot simulation using a polling initialization technique is proposed to reduce the bias by providing a 'good' initial state that is representative of the steady-state condition.

Regenerative simulation is an appealing alternative for parallel simulation because of its potential in producing unbiased results. The significance of the selection of the regeneration state is discussed. A heuristic pilot simulation is proposed for finding a frequently occurring regeneration state using which can achieve short simulation completion time and a large amount of parallelism.

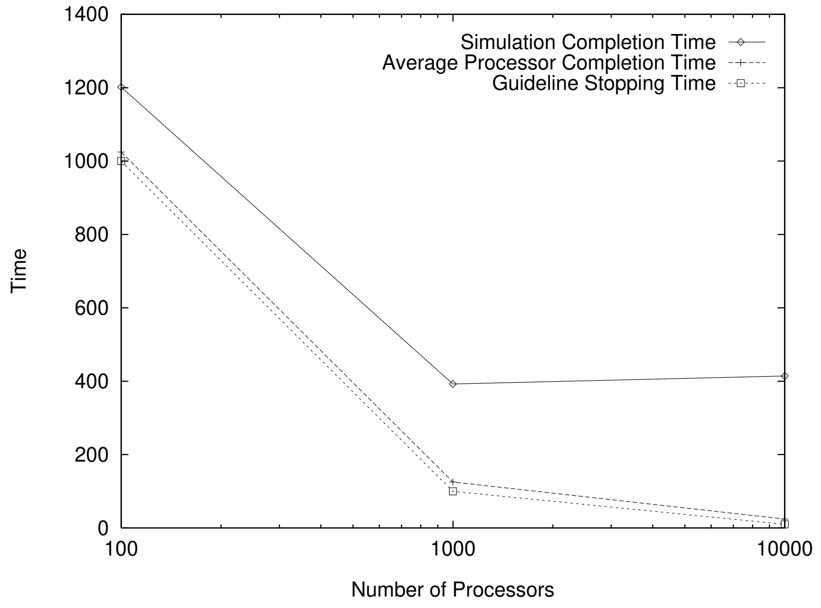

Figure 11: Completion times of the PR simulation for the VC model.

A problem in exploiting regeneration for many applications of interest is that a regeneration state that has short regeneration cycles may not exist. This restricts the applicability of regenerative simulation. To deal with this problem, the idea of allowing approximate regeneration have been introduced in [4, 10]. In this paper, we present an approximate regeneration technique (i.e., PR) which uses substate matching to increase the number of observations. Our experiments with some queueing models show that the PR approach has better statistical efficiency comparing to the MR-D and the PI approaches. The percent error of the PR approach is negligible in almost all cases run.

Note that in many cases, the problems of long expected regeneration intervals can not be solved by the substate matching technique along. For example, in the G/G/1 queue model, when $\lambda/\mu \to 1$, the system becomes unstable and some processors may take a long time to reach partial regeneration. Also, when the essential set W consists of a continuous state variable, a matching substate that results in a finite expected PRCL may not exist. In such cases, we may further relax the regeneration condition by using a state aggregation technique [4], in which we group all possible states corresponding to W into some finite aggregated states. Then a partial regeneration is considered to be reached whenever the system partially regenerates on any substate in a pre-selected aggregated state.

A major disadvantage of the PR approach (as well as any other regenerative simulation approach) is that the simulation completion time is a random variable. Therefore, the execution time of the simulation can not be accurately predicted before the simulation. Another concern of applying the PR approach is the assessment of the simulation accuracy. This can be achieved by conducting a pilot simulation and then evaluating the correlation among the partial regeneration cycles generated by the pilot runs. The amount of the correlation provides an indication of the accuracy of the resultant PR simulation.

Acknowledgment

We would like to thank Philip Heidelberger for his suggestions on this work.

References

[1] Bhavsar V. C., Isaac J. R., Design and Analysis of Parallel Monte Carlo Algorithms. *SIAM J. Sci. Stat. Comput.* 8, (1987), s73-s95.

[2] Chandy, K.M., Sherman, R., Space-Time and Simulation. *Proceedings of the 1989 SCS Multiconference on Distributed Simulation* (March 1989), 53-57.

[3] Crane, M., A., Iglehart, D. L., Simulating Stable Stochastic Systems, III: Regenerative Process and Discrete-Event Simulations. *Operations Research*, 23, 1, (Jan. 1975), 33-45.

[4] Crane, M., A, Iglehart D. L., Simulating Stable Stochastic Systems, IV: Approximation Techniques. *Management Science*, 21, 11, (July 1975), 1215-1244.

[5] David, H. A., *Order Statistics*, 2nd ed., John Wiley, New York, 1981.

[6] Fujimoto, R. M., Parallel Discrete Event Simulation. *Commun. ACM* 33, 10 (Oct. 1990), 31-53.

[7] Glynn, P., Heidelberger, P., Analysis of Parallel Replicated Simulations Under a Completion Time Constraint. *ACM TOMACS.* 1(1), (Jan. 1991), 3-23.

[8] Glynn, P., Heidelberger, P., Analysis of Initial Transient Deletion for Parallel Steady-State Simulations. *SIAM J. Stat. Comput.* 14, 4, (July 1992), 904-922.

[9] A.G. Greenberg, B.D. Lubachevsky and I. Mitrani, Unboundedly Parallel Simulations via Recurrence Relations, *Proc. of the Conf. on Measurement and Modeling of Computer Systems* (1990), pp. 1-12.

[10] Gunther, F., L., Wolff, R. W., The Almost Regenerative Method for Stochastic System Simulation. ORC 75-21, University of California, Berkeley, 1975.

[11] Heidelberger, P., Discrete Event Simulations and Parallel Processing: Statistical Properties. *SIAM J. Stat. Comput.* 9, 6, (Nov. 1988), 1114-1132.

[12] Kelton, W. D., Law, A. M., A New Approach for Dealing with the Startup Problems in Discrete Event Simulation. Naval Res. Logist. Quart., 30,(1983) 641-658.

[13] Kelton, W. D., Random Initialization Methods in Simulation. *IIE Trans*, 21, (1989), 335-367.

[14] Law, A. M., Carson, J. S., A Sequential Procedure for Determining the Length of a Steady-State Simulation. *Operations Research*, 27, (1979), 1011-1025.

[15] Lin, Y. B., Lazowska, E., A Time-Division Algorithm for Parallel Simulation. *ACM TOMACS* 1, 1 (Jan. 1991), 73-83.

[16] Miller, R. G., The Jackknife-A Review. *Biometrika*, 61, (1974), 1-15.

[17] Schwartz, M., Telecommunication Networks Protocols, Modeling and Analysis. Addison Wesley, 1987.

[18] Wang, J., Abrams, M. Determining Initial States of Time-Parallel Simulations. *Proceedings of the 1993 Workshop on Parallel and Distributed Simulation* (May 1993), 19-26.

[19] Wang, J., Abrams, M. Massively Time-Parallel, Approximate Simulation of Loss Queueing Systems. *Annals of the Operations Research,* Vol. 53, (1994), 553-576.