Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

Finite-Horizon and Infinite-Horizon Statistics

Università degli studi di Roma Tor Vergata
Department of Civil Engineering and Computer Science Engineering

1

Algorithm 1.2: using the resulting model

7. **Design simulations experiments**
   - What parameters should be varied?
   - perhaps many combinatoric possibilities
8. **Make production runs**
   - Record initial conditions, input parameters
   - Record statistical output
9. **Analyze the output**
   - Random components → statistical analysis
     (means, standard deviations, percentiles, histograms etc.)
10. Make decisions
    - The step9 results drive the decisions → actions
    - Simulation should be able to correctly predict the outcome of these actions (→ further refinements)
11. Document the results
    - summarize the gained insights in specific observations and conjectures useful for subsequent similar system models

2

Central limit theorem

If $X_1, X_2, \ldots, X_n$ is an iid sequence of random variables (RVs) with

- common mean μ
- common standard deviation σ

and if $\bar{X}$ is the (sample) mean of these RVs

then $\bar{X}$ approaches a *Normal*(μ, $\sigma / \sqrt{n}$) as $n \to \infty$

Theorem 2

If $x_1, x_2, \ldots, x_n$ is an independent random sample from a "source" of data with unknown mean μ, if $\bar{x}$ and s are the mean and standard deviation of this sample, and if n is large, it is approximately true that

$$t = \frac{\bar{x} - \mu}{s / \sqrt{n-1}}$$

is a *Student*($n-1$) random variate

Theorem 3

If $x_1, x_2, \ldots, x_n$ is an independent random sample from a "source" of data with unknown mean μ

- if $\bar{x}$ and s are the sample mean and sample standard deviation
- n is large

Then, given a confidence parameter α with $0.0 < \alpha < 1.0$, there exists an associated positive real number t^* such that

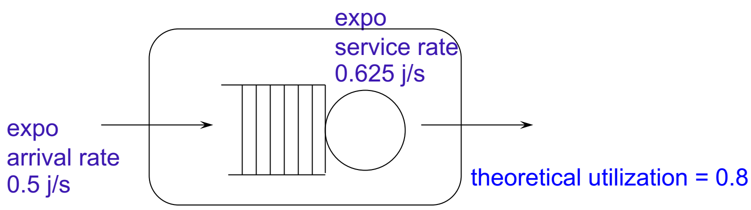
$$Pr\left(\bar{x} - \frac{t^* s}{\sqrt{n-1}} \le \mu \le \bar{x} + \frac{t^* s}{\sqrt{n-1}}\right) \cong 1 - \alpha$$

Algorithm

To calculate an interval estimate for the unknown mean μ of the population from which a random sample $x_1, x_2, \ldots, x_n$ was drawn:

- pick a level of confidence $1-\alpha$ (tipically α =0.05)
- calculate the sample mean $\bar{x}$ and standard deviation s (use Welford's algorithm)
- calculate the critical value t^* = *idfStudent*($n-1$, $1-\alpha/2$)
- calculate the interval endpoints $\bar{x} \pm \frac{t^* s}{\sqrt{n-1}}$

If n is sufficiently large, then you are ($1-\alpha$)x100% confident that the mean μ lies within the interval. The midpoint of the interval is $\bar{x}$



STOP=200
seed (9 digits or less) >> 1234

for 103 jobs
average interarrival time = 1.93
average wait .... = 7.34
average delay .... = 5.60
average service time .... = 1.74
average # in the node ... = 3.69
average # in the queue .. = 2.81
utilization .... = 0.87
jobs remaining in the node= 0

seed (9 digits or less) >> 5678

for 89 jobs
average interarrival time = 2.22
average wait .... = 4.19
average delay .... = 2.61
average service time .... = 1.58
average # in the node ... = 1.88
average # in the queue .. = 1.17
utilization .... = 0.71
jobs remaining in the node= 0


```
....
while ((t.arrival < STOP) || (number > 0)) {
   t.next= Min(t.arrival, t.completion); /* next event time   */
   ....


printf(" … jobs", index);
printf(" average interarrival time ..", t.last / index);
printf(" average wait …", area.node / index);
printf(" average delay ...", area.queue / index);
printf(" average service time ...", area.service / index);
printf(" average # in the node ... ", area.node / t.current);
printf(" average # in the queue .. ", area.queue / t.current);
printf(" utilization ....", area.service / t.current);
printf(" jobs remaining in the node= ....", number);
```

STOP=200
seed (9 digits or less) >> 1234

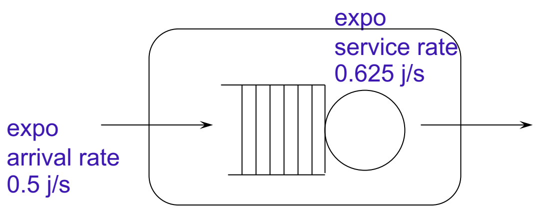
for 103 jobs
average interarrival time = 1.93
average wait .... = 7.34
average delay .... = 5.60
average service time .... = 1.74
average # in the node ... = 3.69
average # in the queue .. = 2.81
utilization .... = 0.87
jobs remaining in the node= 0

for 99 jobs
average interarrival time = 2.01
average wait .... = 7.54
average delay .... = 5.79
average service time .... = 1.75
average # in the node ... = 3.76
average # in the queue .. = 2.89
utilization .... = 0.87
jobs remaining in the node= 4

seed (9 digits or less) >> 5678

for 89 jobs
average interarrival time = 2.22
average wait .... = 4.19
average delay .... = 2.61
average service time .... = 1.58
average # in the node ... = 1.88
average # in the queue .. = 1.17
utilization .... = 0.71
jobs remaining in the node= 0

for 88 jobs
average interarrival time = 2.25
average wait .... = 4.23
average delay .... = 2.64
average service time .... = 1.59
average # in the node ... = 1.88
average # in the queue .. = 1.17
utilization .... = 0.71
jobs remaining in the node= 1



seeds

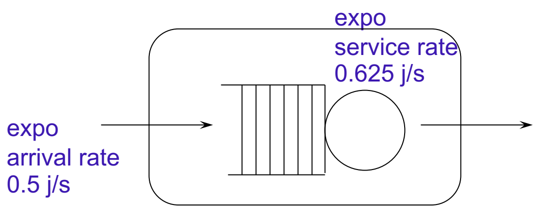
1234	5678	4321
7.34	4.19	2.65
7.54	4.23	2.68

average wait
theoretical value
8 s

STOP=200
about 100 jobs

If the program is executed multiple times varying only the rngs initial seed from replication to replication,

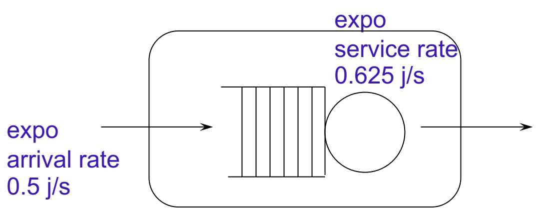
- the average wait in the node will vary significantly
- for most replications, the average wait will not be close to the steady-state average wait
- the initial conditions affect the results



average wait theoretical value 8 s

	seeds			
	1234	**5678**	**4321**	
	7.34	4.19	2.65	STOP=200 about 100 jobs
	7.54	4.23	2.68	
	8.39	7.30	8.59	STOP=20000 about 10000 jobs
	8.34	7.30	8.59	

- If a relatively large number of jobs (say 10000) are used, variability of the average wait will be much less significant and be close to the steady-state average number
- When the number of jobs becomes infinite, the initial condition bias disappears



If we further decrease the service rate → 0.5 j/s, utilization → 1

$\overline{w}, \overline{l}$ will tend to increase, without limit

The server cannot keep up with demand, so the queue will grow as more jobs arrive
In this case, the steady-state average wait and the average number in the node are both infinite

Def. *Steady-state statistics*
Steady-state system statistics are those statistics, if they exist, that are produced by simulating the operation of a *stationary* discrete-event system for an effectively infinite length of time

A *finite-horizon* discrete-event simulation is one for which the simulated operational time is finite

An *infinite-horizon* discrete-event simulation is one for which the simulated operational time is effectively infinite

- Transient system statistics are those statistics that are produced by a finite-horizon discrete-event simulation
- Steady-state statistics are produced by an infinite-horizon simulation
- The initial conditions affect finite-horizon statistics
- The initial conditions do not affect infinite-horizon statistics: after enough time, the system loses memory of its initial state

Another Important Distinction

- In an infinite-horizon simulation, the system "environment" is assumed to remain static
  If the system is a single-server service node, both the arrival rate and the service rate are assumed to remain constant in time
- In a finite-horizon simulation, no need to assume a static environment

Relative Importance of Two Statistics

- The "traditional" view: steady-state statistics are most important
  - Steady-state statistics are better understood because they are much more easy to analyze mathematically
  - It is frequently difficult to accurately model initial conditions and non-stationary system parameters
- The "pragmatic" view: transient statistics are most important because steady-state is just a convenient fiction
- Depending on the application, both transient and steady-state statistics may be important

Relative Importance of Two Statistics

- Important to decide which statistics best characterize the system's performance

one of the most important skills:
the ability to decide, on a system-by-system basis, which kind of statistics best characterizes the system's performance.

Steady-state or Transient Statistics

Consider a bank that opens at 9 AM and closes at 5 PM.
A finite-horizon simulation over the 8-hour period produces transient statistics valuable in determining the optimal staffing of tellers throughout the day.

Consider a fast food restaurant with a drive-up window that experiences a lunch rush period between 11:45 AM and 1:15 PM with an arrival rate that remains constant over the rush period.
This 90-minute period could be simulated for a much longer time period, producing steady-state statistics which might be valuable for estimating the average wait time at the drive-up window.

Initial and Terminal Conditions

- Finite-horizon discrete-event simulations are also known as *terminating simulations*
  - In program ssq4, the *system state* is idle at the beginning and at the end of the simulation
  - The *terminal condition* is specified by the "close the door" time
  - The *system state* of sis4 is the current and on-order inventory levels; these states are the same at the beginning and at the end of the simulation
  - The *terminal condition* is specified by the number of time intervals
- Infinite-horizon discrete-event simulations (*non-terminating simulations*) must be terminated; typically done using whatever stopping conditions are most convenient
  - The steady-state statistics are based on such a huge amount of data that a few "non-steady-state" data points accumulated at the beginning and the end of the simulation should have no significant impact (bias) on output statistics

Formal Representation

- The state variable $X(\cdot)$ is known formally as a *stochastic process*
  - The typical objective of a finite-horizon simulation of this system would be to estimate the time-averaged *transient* statistic

random variable

$$\bar{X}(\tau) = \frac{1}{\tau}\int_0^\tau X(t)dt$$

where $\tau > 0$ is the terminal time

  - The typical objective of an infinite-horizon simulation of this system would be to estimate the time-averaged *steady-state* statistics

is not a random variable

$$\bar{x} = \lim_{\tau\to\infty} \bar{X}(\tau) = \lim_{\tau\to\infty} \frac{1}{\tau}\int_0^\tau X(t)dt$$

Replication

- If a discrete-event simulation is repeated, varying only the rngs initial states from run to run, each run of the simulation program is a *replication* and the totality of replications is said to be *ensemble*
- Replications are used to generate *estimates* of the same transient statistic
- The initial seed for each replication should be chosen to be no replication-to-replication overlap
- The standard way is to use the final state of each rngs stream from *one* replication as the initial state for the *next* replication accomplished by calling PlantSeeds once outside the main replication loop

Independent Replications and Interval Estimation

Suppose the finite-horizon simulation is replicated n times, each time generating a state time history $x_i(t)$

$$\bar{x}_i(\tau) = \frac{1}{\tau}\int_0^\tau x_i(t)dt$$

where $i = 1, 2, \ldots, n$ is the replication index

Each data point $\bar{x}_i(\tau)$ is an independent observation of the random variable $\bar{X}(\tau)$

If n is large enough, the pdf of $\bar{X}(\tau)$ can be estimated from a histogram of the $\bar{x}_i(\tau)$

Independent Replications and Interval Estimation

In practise, it is usually only the expected value $E[\bar{X}(\tau)]$ that is desired.
A *point* estimate of this transient statistic is available as an *ensemble average*, even if n is not large

$$\frac{1}{n}\sum_{i=1}^{n}\bar{x}_i(\tau)$$

An interval estimate for $E[\bar{X}(\tau)]$ can be calculated

Use the interval estimation technique from Section 8.1

Example 8.3.6

A modified version of ssq2 was used to produce 20 replications

- 100 jobs processed through M/M/1 service node
  - Node is initially idle
  - Arrival rate is = 1.0
  - Service rate is = 1.25
- The resulting 20 observations of the average wait in the node:

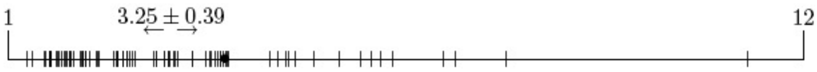
from program estimate 95%-confidence-interval: we are 95% confident that if we were to do millions of replications the ensamble average would be somewhere between 2.29 and 4.37

4 is the exact steady state value

It is possible that 100 jobs are enough to produce steady-state statistics

Example 8.3.7

- The modified version of program ssq2 was used to produce 60 more replications
- Consistent with $\sqrt{n}$ rule, expect two-fold decrease in the width of the interval estimate
- Based on 80 replications, the resulting 95% confidence interval estimate was 3.25 ± 0.39 (2.86, 3.64)



In this case 100 jobs <u>are not enough</u> to produce steady-state statistics

the bias of the initially idle state is still evident in the transient statistic

Example 8.3.8

- As a continuation of Example 8.3.6, the number of jobs per replication was increased from 100 to 1000
- 20 replications were used to produce 20 observations of the average wait in the node (3.45, 4.19)

Relative to Example 8.3.6, much more symmetric sample mean in Example 8.3.8 (2.29, 4.37)

Example 8.3.8

- The 1000-jobs per replication results are more consistent with the underlying theory of interval estimation
  - Requires a sample mean distribution that is approximately *Normal*(μ, $\sigma / \sqrt{n}$)
  - Sample mean distribution is centered on (unknown) population
- 1000 jobs may achieve steady-state; 100 jobs cannot