EEC 686/785
Modeling & Performance Evaluation of Computer Systems

Lecture 15

Wenbing Zhao
Department of Electrical and Computer Engineering
Cleveland State University
wenbing@ieee.org
(based on Dr. Raj Jain's lecture notes)

Outline

- Review of lecture 14
- Analysis of simulation results
  - Model verification techniques
  - Model validation techniques
  - Transient removal
  - Terminating simulations
  - Stopping criteria: variance estimation
  - Variance reduction

Terminology

- State variables
- Event: a change in the system state
- Continuous-time and discrete-time modes
- Continuous state and discrete state models
- Deterministic and probabilistic models
- Static and dynamic models
- Linear and nonlinear models
- Open and closed models
- Stable and unstable models

Types of Simulations

- Emulation: using hardware or firmware
  - E.g., terminal emulator, processor emulator
  - Mostly hardware design issues
- Monte Carlo simulation
- Trace-driven simulation
- Discrete event simulation
- Process-oriented simulation

Monte Carlo Simulation

- Static simulation (no time axis)
- To model probabilistic phenomenon
- Need pseudorandom numbers
- Used for evaluating nonprobabilistic expressions using probabilistic methods

Trace-Driven Simulation

- Trace = time ordered record of events on a system
- Trace-driven simulation = a simulation using a trace as its input
- Used in analyzing or tuning resource management algorithms
  - Paging, cache analysis, CPU scheduling, deadlock prevention, dynamic storage allocation

Components of Discrete Event Simulations

- Event scheduler
- Simulation clock and a time-advancing mechanism
- System state variables
- Event routines
- Input routines
- Report generator
- Initialization routines
- Trace routines
- Dynamic memory management
- Main program

Process-Oriented Simulation

- Event-driven simulation
  - Strength: it is general => any discrete-event simulation may be implemented
  - Weakness: it is not user-friendly in some respects
    - Each event stands alone <= by definition events maintain no context because they only exist for zero simulation time
- Process-oriented simulation: allows related state changes to be combined in the context of a process

Process-Oriented Simulation

- A sets of related event types are grouped together in a process, similar to a process in an operating system
- A process consists of body of code, resources allocated and state
- The simulation driver is more complicated: in addition to managing the event list, it has to manage
  - process creation and activation/scheduling
  - process suspension and process termination
  - interprocess communication/synchronization

Analysis of Simulation Results

- Analysis of simulation results
- Model verification techniques
- Model validation techniques
- Transient removal
- Terminating simulations
- Stopping criteria: variance estimation
- Variance reduction

Model Verification vs. Validation

- **Verification** => debugging, correct implementation of the model
  Validation => model = real world, valid assumption
- Four possibilities
  - Unverified, invalid
  - Unverified, valid
  - Verified, invalid
  - Verified, valid

Model Verification Techniques

- Top down modular design
- Antibugging
- Structured walk-through
- Deterministic models
- Run simplified cases
- Trace
- Online graphic displays
- Continuity test
- Degeneracy tests
- Consistency tests
- Seed independence

Top Down Modular Design

- Divide and conquer
- Modules = subroutines, subprograms, procedures
  - Modules have well defined interfaces
  - Can be independently developed, debugged, and maintained
  - Top-down design => hierarchical structure => modules and submodules

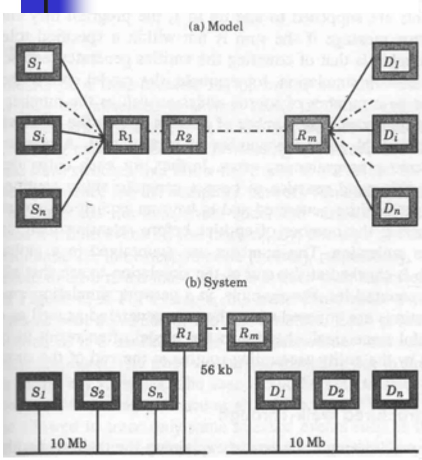
Model Verification Techniques

- **Antibugging**: include self-checks, analogous to assertions in C/C++
  - $\sum$ probabilities = 1
  - Jobs left = generated – serviced
- **Structured walk-through**
  - Explain the code to another person or group
  - Works even if the person is sleeping
- **Deterministic models**: use constant values / distributions
- **Run simplified cases**
  - Only one packet, only one source, only one intermediate node

Trace

- Trace = time-ordered list of events and variables
  - Several levels of detail
    - Events trace, procedure trace, variables trace
  - User selects the detail
  - Include on and off switch

Trace



PKT=100 CR=4 #S=2 RS=2 — !This was a run with 100 packets, 4 credits per source, two sources. Router service time was twice that of the source.

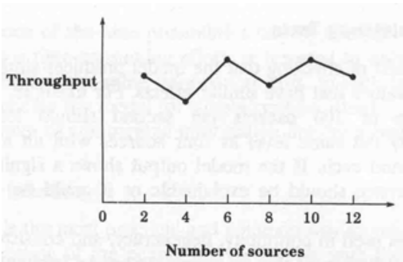
Time	Node: Event	Pkt #/ Attempt	Sample Delay	Delay Estimate	
0.00:	S1: TIMR	1- 1			!Round trip delay-measuring stop watch started by source 1
	S1: SEND	1- 1/ 1			!Packet 1 sent by source 1
	S1: STRT				!Timeout alarm clock set by source 1
	S2: TIMR	2- 1			!Round trip delay-measuring stop watch started by source 2
	S2: SEND	2- 1/ 1			!Packet 1 sent by source 2
	S2: STRT				!Timeout alarm clock set by source 2
1.00:	R1: QUED	1- 1/ 1			!Packet 1 of source 1 was put into a queue at router 1
	R1: LOST!	2- 1/ 1			!Packet 1 of source 2 was lost due to lack of buffer at router 1
3.00:	D1: RECD	1- 1/ 1			!Packet 1 of source 1 was received at destination 1
	S1: ACKD	1- 1			!Acknowledgment for packet 1 was received at source 1
	S1: UPDT	1- 1	3.00	3.00	!Source 1 updated its estimate of round trip delay

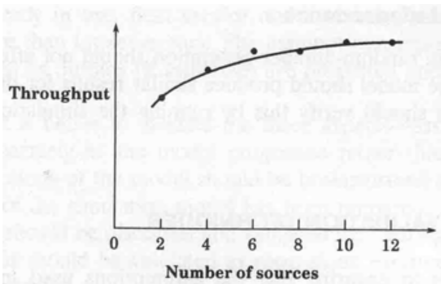
Online Graphic Displays

- Make simulation interesting
- Help selling the results
- More comprehensive than trace

Continuity Test

- Run for different values of input parameters
- Slight change in input => slight change in output





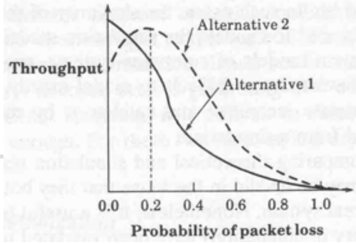
More Verification Techniques

- **Degeneracy tests**
  - Try extreme configuration and workloads
  - One CPU, zero disk
- **Consistency tests**
  - Similar result for inputs that have same effect
  - Four users at 100 Mbps vs. two at 200 Mbps
- **Seed independence**: similar results for different seeds

Model Validation Techniques

- Validation techniques for one problem may not apply to another problem
- Aspects to validate
  - Assumptions
  - Input parameter values and distributions
  - Output values and conclusions
- Techniques
  - Expert intuition
  - Real system measurements
  - Theoretical results
- => $3 \times 3 = 9$ validation tests

Expert Intuition



- Most practical and common way
- Experts = involved in design, architecture, implementation, analysis, marketing, or maintenance of the system
- Selection = function of life cycle stage
- Present assumption, input, output
- Better to validate one at a time
- See if the experts can distinguish simulation vs. measurement

Real System Measurements

- Compare assumptions, input, output with the real world
- Often infeasible or expensive
- Even one or two measurements add to the validity

Theoretical Results

- Analysis = simulation
- Used to validate analysis also
- Both may be invalid
- Use theory in conjunction with experts' intuition
  - E.g., use theory for a large configuration
- "Fully validated model" is a myth

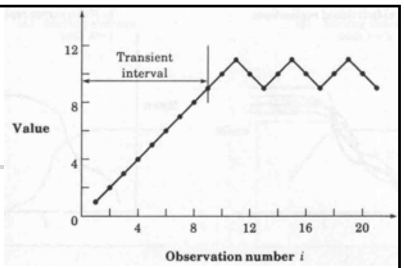
Transient Removal

- General steady state performance is interesting
  - Remove the initial part. No exact definition => heuristics
- Transient removal methods
  - Long runs
  - Proper initialization
  - Truncation
  - Initial data deletion
  - Moving average of independent replications
  - Batch means

Transient Removal Techniques

- **Long runs**: so that the transient phase become statistically unimportant
  - Wastes resources
  - Difficult to insure that it is long enough
- **Proper initialization**
  - Start in a state close to expected steady state => reduces the length and effect of transient state

Truncation



- Assumes variability is lower during steady state
- Plot max-min of *n*-*l* observation for *l*=1,2,…
- When (*l*+1)th observation is neither the minimum nor maximum => transient state ended
- Example: 1,2,3,4,5,6,7,8,9,10,11,10,9,10,11,10,9,10,11,10,9,… At *l*=9, range = (9,11), next observation = 10
  - Sometimes incorrect result

Initial Data Deletion

- Delete some initial observation
- Compute average
- No change => steady state
- Use several replications to smoothen the average
- *m* replications of size *n* each x_{ij} = *j*th observation in the *i*th replication

Initial Data Deletion - Steps

- 1. Get a mean trajectory by averaging across replications: $\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij} \quad j = 1,2,\ldots,n$
- 2. Get the overall mean: $\bar{\bar{x}} = \frac{1}{n}\sum_{j=1}^{n} \bar{x}_j$

  set *l*=1 and proceed to the next step
- 3. Delete the first *l* observations and get an overall mean from the remaining *n-l* values:

$$\bar{\bar{x}}_l = \frac{1}{n-l}\sum_{j=l+1}^{n} \bar{x}_j$$

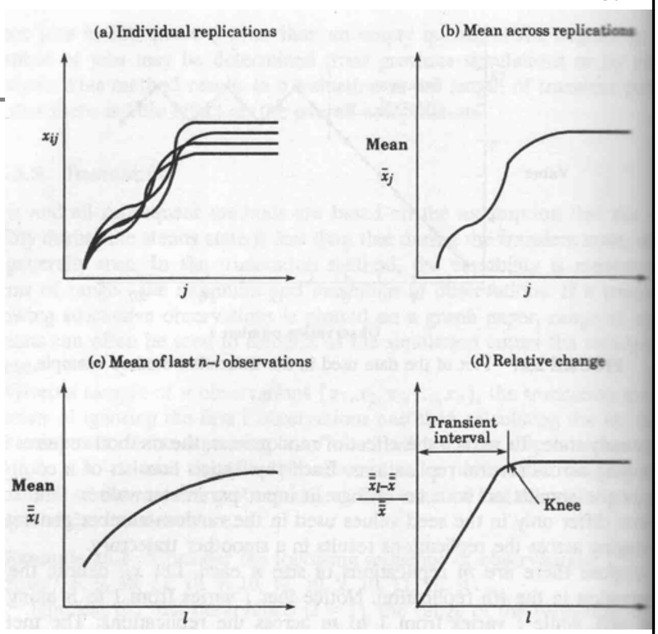
Initial Data Deletion - Steps

- 4. Compute the relative change:

$$\text{Relative change} = \frac{\bar{\bar{x}}_l - \bar{\bar{x}}}{\bar{\bar{x}}}$$

- 5. Repeat steps 3 and 4 by varying l from 1 to n-1
- 6. Plot the overall mean and the relative change
- 7. l at knee = length of the transient interval

Initial Data Deletion



Moving Average of Independent Replications

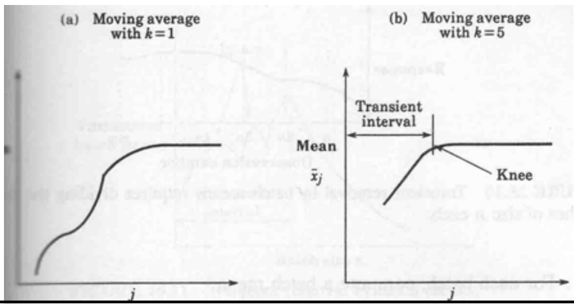
- Mean over a moving time interval window
  - 1. Get a mean trajectory by averaging across replications: $\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij} \quad j = 1,2,\ldots,n$

    Set k=1 and proceed to the next step
  - 2. Plot a trajectory of the moving average of successive 2k+1 values:

$$\bar{\bar{x}}_j = \frac{1}{2k+1}\sum_{l=-k}^{k} \bar{x}_{j+l} \quad j = k+1, k+2, \ldots, n-k$$

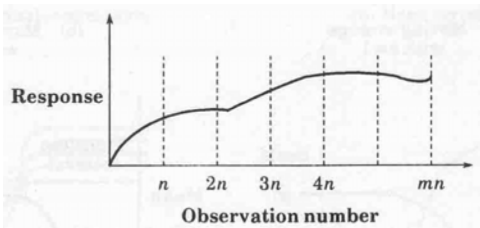
Moving Average of Independent Replications

- Mean over a moving time interval window
  - 3. Repeat step 2, with k=2,3, and so on until the plot is smooth
  - 4. Value of j at the knee gives the length of the transient phase



Batch Means

- Run a long simulation and divide into equal duration part
- Part = batch = subsample
- Study variance of batch means as a function of the batch size



Batch Means - Steps

- 1. For each batch, compute a batch mean:

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \quad i = 1,2,\ldots,m$$

- 2. Compute overall mean:

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m} \bar{x}_i$$

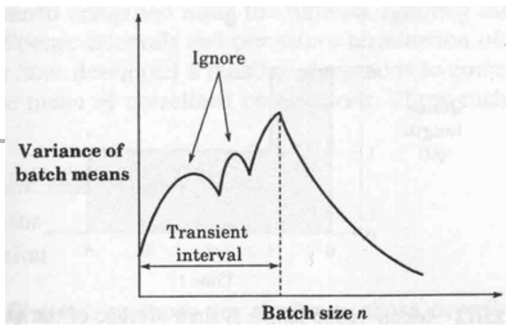
- Compute the variance of the batch means:

$$Var(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m} (\bar{x}_i - \bar{\bar{x}})^2$$

Batch Means - Steps

- 4. Repeat steps 1 and 3, for *n*=3,4,5, and so on
- 5. Plot the variance as a function of batch size *n*
- 6. Value of *n* at which the variance definitely starts decreasing gives transient interval

Batch Means



- Rationale:
  - Batch size « transient => overall mean = initial mean => lower variance
  - Batch size » transient => overall mean = steady state mean => lower variance
  - Ignore peaks followed by an upswing

Terminating Simulations

- Some times, do not need transient removal:
  - Transient performance is of interest, e.g., network traffic
  - System shuts down
- Final conditions:
  - May need to exclude the final portion from results
  - Techniques similar to transient removal

Treatment of Leftover Entities

- Mean service time:

$$= \frac{\text{Total service time}}{\text{Number of jobs that completed service}}$$

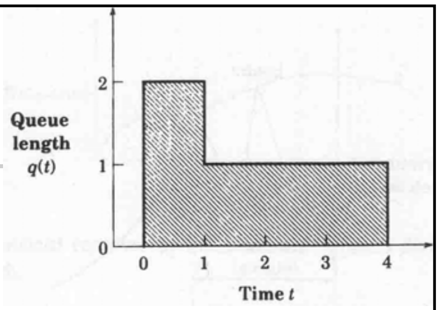
- Mean waiting time:

$$= \frac{\text{Sum of waiting time}}{\text{Number of jobs that received service}}$$

- Mean queue length:

$$\neq \frac{\sum_{j=1}^{n} \text{Queue length at event } j}{\text{Number of events } n} = \frac{1}{T}\int_0^T \text{Queue\_length}(t)dt$$

Example



- Mean of queue length
- 3 events
  - 2 jobs arrival at t=0
  - 1st job departs at t=1
  - 2nd job departs at t=4
- Mean queue length ≠ (2+1+0)/3
- Mean queue length = area under queue length curve / duration = 5/4 = 1.25

Stopping Criteria: Variance Estimation

- Run until confidence interval is narrow enough

$$\bar{x} \pm z_{1-\alpha/2}\sqrt{Var(\bar{x})}$$

- For independent observations:

$$Var(\bar{x}) = \frac{Var(x)}{n}$$

- Independence not applicable to most simulations. Large waiting time for *i*th job => large waiting time for (*i*+1)th job

Stopping Criteria: Variance Estimation

- For correlated observations:

  Actual variance » Var(x)/n
- Solutions:
  - Independent replications
  - Batch means
  - Method of regeneration

Independent Replications

- Assumes that means of independent replications are independent
- Conduct m replications of size $n+n_0$ each
  - 1. Compute a mean for each replications:

$$\bar{x}_i = \frac{1}{n}\sum_{j=n_0+1}^{n_0+n} x_{ij} \quad i = 1,2,\ldots,m$$

  - 2. Compute an overall mean for all replications:

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m} \bar{x}_i$$

Independent Replications

- 3. Calculate the variance of replicate means:

$$Var(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

- 4. Confidence interval for the mean response is:

$$\left[\bar{\bar{x}} \mp z_{1-\alpha/2}\sqrt{Var(\bar{x})}\right]$$

- Keep replications large to avoid waste. Ten replications generally sufficient

Batch Means

- Also called method of subsamples
- Run a long simulation run. Discard initial transient interval, and divide the remaining observations run into several batches or subsamples
  - 1. Compute means for each batch:

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \quad i = 1,2,\ldots,m$$

  - 2. Compute an overall mean:

$$\bar{\bar{x}} = \frac{1}{m}\sum_{i=1}^{m}\bar{x}_i$$

Batch Means

- 3. Calculate the variance of batch means:

$$Var(\bar{x}) = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{x}_i - \bar{\bar{x}})^2$$

- 4. Confidence interval for the mean response is:

$$\left[\bar{\bar{x}} \mp z_{1-\alpha/2}\sqrt{Var(\bar{x})}\right]$$

- Less waste than independent replications
- Keep batches long to avoid correlation
- Check: compute the autocovariance of successive batch means:

$$Cov(\bar{x}_i, \bar{x}_{i+1}) = \frac{1}{m-2}\sum_{i=1}^{m-1}(\bar{x}_i - \bar{\bar{x}})(\bar{x}_{i+1} - \bar{\bar{x}})$$

Double *n* until autocovariance is small

Case Study: Interconnection Networks

- Indirect binary n-cube networks: used for processor-memory interconnection. Two stage network with full fanout
- At 64, autocovariance < 1% of sample variance

Batch Size	Autocovariance	Variance
1	-0.18792	1.79989
2	0.02643	0.81173
4	0.11024	0.42003
8	0.08979	0.26437
16	0.04001	0.17650
32	0.01108	0.10833
64	0.00010	0.06066
128	-0.00378	0.02992
256	0.00027	0.01133
512	0.00069	0.00503
1024	0.00078	0.00202