Simulation

Performance Modeling Lecture #8

Slides adapted from Mark Claypool

Introduction (1 of 3)

The best advice to those about to embark on a very large simulation is often the same as Punch's famous advice to those about to marry: 'Don't!'
– Bratley, Fox and Schrage (1986)

- System to be characterized may not be available
  - During design or procurement stage
- Still want to predict performance
- Or, may have system but want to evaluate wide-range of workloads
  → Simulation
- However, simulations may fail
  - Need good programming, statistical analysis and perf eval knowledge

Outline

- Introduction
- Common Mistakes in Simulation
- Terminology
- Types of Simulations
- Verification and Validation

Common Mistakes in Simulation (1 of 4)

- *Inappropriate level of detail*
  - Level of detail often potentially unlimited
  - But more detail requires more time to develop
    - And often to run!
  - Can introduce more bugs, making more inaccurate not less!
  - Often, more detailed viewed as "better" but may not be the case
    - More detail requires more knowledge of input parameters
    - Getting input parameters wrong may lead to more inaccuracy (Ex: disk service times exponential vs. simulating sector and arm movement)
  - Start with less detail, study sensitivities and introduce detail in high impact areas

Common Mistakes in Simulation (2 of 4)

- *Improper language*
  - Choice of language can have significant impact on time to develop
  - Special-purpose languages can make implementation, verification and analysis easier
  - C++Sim (http://cxxsim.ncl.ac.uk/), JavaSim (http://javasim.ncl.ac.uk/), SimPy(thon) (http://simpy.sourceforge.net/) …
- *Unverified models*
  - Simulations generally large computer programs
  - Unless special steps taken, bugs or errors

Common Mistakes in Simulation (3 of 4)

- *Invalid models*
  - No errors, but does not represent real system
  - Need to validate models by analytic, measurement or intuition
- *Improperly handled initial conditions*
  - Often, initial trajectory not representative of steady state
    - Including can lead to inaccurate results
  - Typically want to discard, but need method to do so effectively

Common Mistakes in Simulation (4 of 4)

- *Too short simulation runs*
  - Attempt to save time
  - Makes even *more* dependent upon initial conditions
  - Correct length depends upon the accuracy desired (confidence intervals)
  - Variance estimates
- *Poor random number generators and seeds*
  - "Home grown" are often not random enough
    - Makes artifacts
  - Best to use well-known one
  - Choose seeds that are different

More Causes of Failure (1 of 2)

> *Any given program, when running, is obsolete. If a program is useful, it will have to be changed. Program complexity grows until it exceeds the capacity of the programmer who must maintain it.* - Datamation 1968

> *Adding manpower to a late software project makes it later.*
> - Fred Brooks

- *Large software*
  - Quotations above apply to software development projects, including simulations
  - If large simulation efforts not managed properly, can fail
- *Inadequate time estimate*
  - Need time for *validation* and *verification*
  - Time needed can often grow as more details added

More Causes of Failure (2 of 2)

- *No achievable goal*
  - Common example is "model X"
    - But there are many levels of detail for X
  - Goals: Specific, Measurable, Achievable, Repeatable
  - Project without goals continues indefinitely
- *Incomplete mix of essential skills*
  - Team needs one or more individuals with certain skills
  - Need: leadership, modeling and statistics, programming, knowledge of modeled system

Simulation Checklist (1 of 2)

- Checks before developing simulation
  - Is the goal properly specified?
  - Is detail in model appropriate for goal?
  - Does team include right mix (leader, modeling, programming, background)?
  - Has sufficient time been planned?
- Checks during simulation development
  - Is random number random?
  - Is model reviewed regularly?
  - Is model documented?

Simulation Checklist (2 of 2)

- Checks after simulation is running
  - Is simulation length appropriate?
  - Are initial transients removed?
  - Has model been verified?
  - Has model been validated?
  - Are there any surprising results? If yes, have they been validated?

Outline

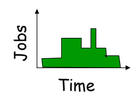
- Introduction
- Common Mistakes in Simulation
- Terminology
- Selecting a Simulation Language
- Types of Simulations
- Verification and Validation
- Transient Removal
- Termination

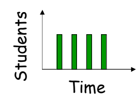
Terminology (1 of 7)

- Introduce terms using an example of simulating CPU scheduling
  - Study various scheduling techniques given job characteristics, ignoring disks, display…
- *State variables*
  - Variables whose values define current state of system
  - Saving can allow simulation to be stopped and restarted later by restoring all state variables
  - Ex: may be length of the job queue

Terminology (2 of 7)

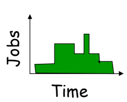
- *Event*
  - A change in system state
  - Ex: Three events: arrival of job, beginning of new execution, departure of job
- *Continuous-time and discrete-time models*
  - If state defined at all times → continuous
  - If state defined only at instants → discrete
  - Ex: class that meets M-F 2-3 is discrete since not defined other times

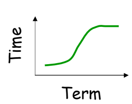




Terminology (3 of 7)

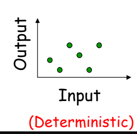
- *Continuous-state and discrete-state models*
  - If uncountably infinite → *continuous*
    - Ex: time spent by students on hw
  - If countable → *discrete*
    - Ex: jobs in CPU queue
  - Note, continuous time does not necessarily imply continuous state and vice-versa
    - All combinations possible

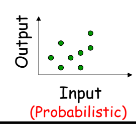




Terminology (4 of 7)

- *Deterministic and probabilistic (stochastic) models*
  - If output predicted with certainty → *deterministic*
  - If output different for different repetitions → *probabilistic*

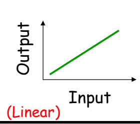


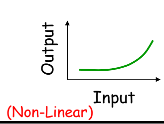


(vertical lines)

Terminology (5 of 7)

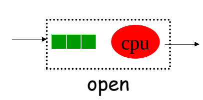
- *Static and dynamic models*
  - Time is not a variable → *static*
  - If changes with time → *dynamic*
  - Ex: CPU scheduler is dynamic, while matter-to-energy model $E=mc^2$ is static
- *Linear and nonlinear models*
  - Output is linear combination of input → *linear*
  - Otherwise → nonlinear

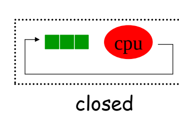




Terminology (6 of 7)

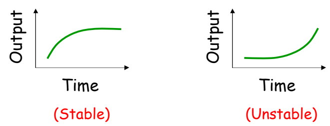
- *Open and closed models*
  - Input is external and independent → *open*
  - Closed model has no external input
  - Ex: if same jobs leave and re-enter queue then closed, while if new jobs enter system then open





Terminology (7 of 7)

- *Stable and unstable*
  - Model output settles down → *stable*
  - Model output always changes → *unstable*

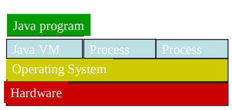


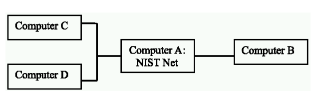
Outline

- Introduction
- Common Mistakes in Simulation
- Terminology
- Types of Simulations
- Verification and Validation

Types of Simulations

- Variety of types, but main: emulation, Monte Carlo, trace driven, and discrete-event
- *Emulation*
  - Simulation that runs on a computer to make it appear to be something else
  - Examples: JVM, NIST Net

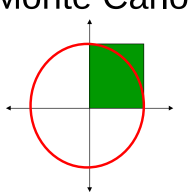




Monte Carlo Simulation (1 of 2)

- A static simulation has no time parameter
  - Runs until some equilibrium state reached
- Used to model physical phenomena, evaluate probabilistic system, numerically estimate complex mathematical expression
- Driven with random number generator
  - So “Monte Carlo” (after casinos) simulation
- Example, consider numerically determining the value of π
- Area of circle = π^2 for radius 1

Monte Carlo Simulation (2 of 2)



- Imagine throwing dart at square
  - Random x (0,1)
  - Random y (0,1)
- Count if inside
  - $\text{sqrt}(x^2+y^2) < 1$
- Compute ratio R
  - in / (in + out)
- Can repeat as many times as needed to get arbitrary precision

- Unit square area of 1
- Ratio of area in quarter to area in square = R
  - $\pi = 4R$

Trace-Driven Simulation

- Uses time-ordered record of events on real system as input
  - Ex: to compare memory management, use trace of page reference patterns as input, and can model and simulate page replacement algorithms
- Note, need trace to be independent of system
  - Ex: if had trace of disk events, could not be used to study page replacement since events are dependent upon current algorithm

Trace-Driven Simulation Advantages

- *Credibility* – easier to sell than random inputs
- *Easy validation* – when gathering trace, often get performance stats and can validate with those
- *Accurate workload* – preserves correlation of events, don't need to simplify as for workload model
- *Less randomness* – input is deterministic, so output may be (or will at least have less non-determinism)
- *Fair comparison* – allows comparison of alternatives under the same input stream
- *Similarity to actual implementation* – often simulated system needs to be similar to real one

Trace-Driven Simulation Disadvantages

- *Complexity* – requires more detailed implementation
- *Representativeness* – trace from one system may not represent all traces
- *Finiteness* – can be long, so often limited by space but then that time may not represent other times
- *Single point of validation* – need to be careful that validation of performance gathered during a trace represents only 1 case
- *Trade-off* – it is difficult to change workload since cannot change trace. Changing trace would first need workload model

Outline

- Introduction
- Common Mistakes in Simulation
- Terminology
- Types of Simulations
- Verification and Validation

Analysis of Simulation Results

> *Always assume that your assumption is invalid.*
> – Robert F. Tatman

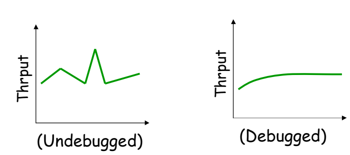
- Would like model output to be close to that of real system
- Made assumptions about behavior of real systems
- 1st step, test if assumptions are reasonable
  - *Validation*, or representativeness of assumptions
- 2nd step, test whether model implements assumptions
  - *Verification*, or *correctness*

Model Verification Techniques (1 of 3)

- Good software engineering practices will result in fewer bugs
- Top-down, modular design
- Assertions (antibugging)
  - Say, total packets = packets sent + packets received
  - If not, can halt or warn
- Structured walk-through
- Simplified, deterministic cases
  - Even if end-simulation will be complicated and non-deterministic, use simple repeatable values (maybe fixed seeds) to debug
- Tracing (via print statements or debugger)

Model Verification Techniques (2 of 3)

- Continuity tests
  - Slight change in input should yield slight change in output, otherwise error



- Degeneracy tests
  - Try extremes (lowest and highest) since may reveal bugs

Model Verification Techniques (3 of 3)

- Consistency tests – similar inputs produce similar outputs
  - Ex: 2 sources at 50 pkts/sec produce same total as 1 source at 100 pkts/sec
- Seed independence – random number generator starting value should not affect final conclusion (maybe individual output, but not overall conclusion)

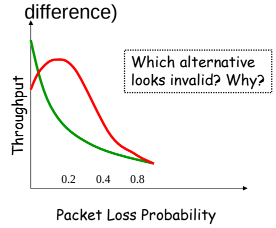
Model Validation Techniques

- Ensure assumptions used are reasonable
  - Want final simulated system to be like real system
- Unlike verification, techniques to validate one simulation may be different from one model to another
- Three key aspects to validate:
  - Assumptions
  - Input parameter values and distributions
  - Output values and conclusions
- Compare validity of each to one or more of:
  - Expert intuition
  - Real system measurements
  - Theoretical results

→ 9 combinations
- Not all are always possible, however

Model Validation Techniques - Expert Intuition

- Most practical, most common
- "Brainstorm" with people knowledgeable in area
- Assumptions validated first, followed soon after by input. Output validated as soon as output is available (and verified), even if preliminary
- Present measured results and compare to simulated results (can see if experts can tell the difference)



Model Validation Techniques - Real System Measurements

- Most reliable and preferred
- May be unfeasible because system does not exist or too expensive to measure
  - That could be why simulating in the first place!
- But even one or two measurements add an enormous amount to the validity of the simulation
- Should compare input values, output values, workload characterization
  - Use multiple traces for trace-driven simulations
- Can use statistical techniques (confidence intervals) to determine if simulated values different than measured values

Model Validation Techniques - Theoretical Results

- Can be used to compare a simplified system with simulated results
- May not be useful for sole validation but can be used to complement measurements or expert intuition
  - Ex: measurement validates for one processor, while analytic model validates for many processors
- Note, there is no such thing as a "fully validated" model
  - Would require too many resources and may be impossible
  - Can only show is invalid
- Instead, show validation in a few select cases, to lend *confidence* to the overall model results

Question

- Imagine you are called in as an expert to review a simulation study. Which of the following would you consider non-intuitive and would want extra validation?
  1. Throughput increases as load increases
  2. Throughput decreases as load increases
  3. Response time increases as load increases
  4. Response time decreases as load increases
  5. Loss rate decreases as load increases