Vorlesung Leistungsanalyse

Simulation Methods and Random Number Generators

Matthias Müller (matthias.mueller@tu-dresden.de)
Bert Wesarg (bert.wesarg@tu-dresden.de)
Simulation Validation and Verification

Matthias Müller (matthias.mueller@tu-dresden.de)
Bert Wesarg (bert.wesarg@tu-dresden.de)
Validation and Verification

- Does the simulation model represents the real system?
  - Or: Are the assumptions about the real system (the system being modeled) reasonable?
    - $\Rightarrow$ validation

- Is the simulation model correctly implemented?
  - Or: Does the simulation implements these assumptions correct?
    - $\Rightarrow$ verification
Model Validation Techniques

Key aspects of a model:

1. Assumptions
2. Input parameter values and distributions
3. Output values and conclusions

Each of these aspects needs to be validated

Three possible sources of validity tests:

1. Expert intuition
2. Real system measurements
3. Theoretical results
Expert Intuition

- Experts may be designers, system architects, implementers, analysts, customers, ...
- Let experts discuss the key aspects
- Integrate these discussion into the software development process
- Repeat these discussions
Real System Measurements

- Most reliable and preferred way to validate the model
- But a real system may not exist or too expensive to carry out
- All three key aspects should be compared with the real system
Theoretical Results

- There are cases where a analytical model is available with simplified assumptions.
- With this model it may also be possible to analytically determine the input distributions.
- Comparing theoretical results and simulation results should be handled with care, because both may be invalid.
Model Verification Techniques

- Also called “debugging”
- Simulation models are large software programs, therefore any techniques known from the field of software engineering are useful to verify a simulation
- Modularity or top-down design
- Antibugging
  - Integrated run-time checking of computed values to detect
- Structured Walk-Through
  - The developer explains the code to another person
Model Verification Techniques (cont’d)

- Deterministic models
  - Disable the distribution of the input variables
- Run simplified cases
  - Results can be easily analyzed
- Trace or online monitoring
- Continuity test
  - Run the simulation multiple times with slightly different input variables
  - Should only result in slightly different output values
- Degeneracy tests
  - Model works for the extreme cases (min and max)
Model Verification Techniques (cont’d)

- Consistency tests
  - The simulation should produce similar results for different input variables that have similar effects

- Seed independence
  - Seeds used in random-number generators should not affect the output
Random Number Generation

Matthias Müller (matthias.mueller@tu-dresden.de)
Bert Wesarg (bert.wesarg@tu-dresden.de)
Random Number Generators (RNG)

- Pseudo-Random (should generate reproducible sequence of pseudo-random numbers);

\[ x_i \]

- Should have well defined distribution (uniformly distributed)

- Fast

- High quality:
  - No correlation \[ \langle x_i x_{i+n} \rangle = \langle x_i \rangle^2 \]
  - Long sequence length
Congruential Generators

Iterative generation of random numbers:

\[ x_{i+1} = (cx_i) \mod p \]

Condition for high quality numbers:

- Theory:
  - \( p \) is prime
  - \( P \) smallest number with:
  \[ c^{p-1} \mod p = 1 \]
- Practice: use literature
  - “Minimal Standard”, Park and Miller
    \[ p = 2^{31} - 1 \]
    \[ c = 7^5 = 16807 \]
Other examples from literature:

- Fishman and Moore (1986):
  \[ p = 2^{31} - 1 \quad \quad \quad c = 48271 \]

- DEC-20 Fortran:
  \[ p = 2^{31} - 1 \quad \quad \quad c = 630360016 \]
N-cube Correlations

- Choose n-tupels from sequence:
  
  $$(x_i, x_{i+1}, ..., x_{i+n}), (x_{i+1} x_{i+2}, ..., x_{i+n+1}), \ldots$$

- Interpret them as points in n-dimensional space.

- Plot them (example for $p=31, c=3$ and $p=31, c=11$):
Lagged-Fibonacci Generators

- Sequence of BINARY numbers
- New number is generated based on previous numbers:

\[ x_0 = \sum_i x_{n_i} \mod 2 \]

- Example:

\[ x_0 = (x_1 + x_2 + x_5 + x_{12}) \mod 2 \]
Kirkpatrick-Stoll Generator

- Use 32 lagged Fibonacci Generators in parallel
- Generation of random numbers:

\[ x_0 = x_{103} + x_{250} \]

- Problems:
  - First 250 numbers have to be generated elsewhere
  - Different sequences have to be uncorrelated

- Advantage:
  - Very long sequence
  - Fast (Bit operations/ additions MOD 2 is XOR)
Other Generators

- Marsaglia (has two seeds)
- Many more in literature
Myths about RNGs

- A complex set of operations leads to a random result
- A single test such as chi-square test is sufficient to test the quality
- RNGs are unpredictable
- Some seeds are better than others (might be true for some RNGs)
- There is one best RNG (remember: today's best is tomorrows second best)
Recommendations for using RNGs

- Do not subdivide one stream (n-cube correlations)
- Consider the periodicity: use non-overlapping streams
- Seed once in your application run, not every time you enter a specific subtask
- Do not use random seeds, or at least document them
- Reproduce your results with different seeds AND different RNGs.

There are famous examples that a “physical property, e.g. in phase transition simulations” was depending on the RNG used.
Tests for RNGs

- Check distribution
- Check mean value
- Check if mean value of all bits is 0.5
- Check n-cube correlations
- Check correlations

\[ \langle x_i x_{i+n} \rangle = \langle x_i \rangle^2 \]

- Power distribution in frequency space (white noise)
- Check if partial sums of a sequence have Gaussian distribution (Chi-square test)
Operational Method

Matthias Müller (matthias.mueller@tu-dresden.de)
Bert Wesarg (bert.wesarg@tu-dresden.de)
Operational Method

- Is based on a set of concepts that correspond naturally and directly to observed properties of real computer systems.

- The computer system will be modeled with the help of a queuing network.

- A queuing network has two types of nodes: wait and delay nodes.

- A wait node consists of a input queue and a server:
  - Jobs arrive at in the input queue.
  - The server can only work one job at the time.
  - A server is not idle when there are jobs in his queue.

- Delay nodes can serve multiple jobs at once:
  - Jobs stay until there are finished.

- Jobs which have fulfilled they total work time demand leave the system.
1. Flow balance in each node
   - Number of arrivals equals the number of leavings is in each node

2. One step at the time
   - At each time in the observation period there is at most one arrival or one leave of a job in the system

3. Routing homogeneity
   - The routing of jobs from one node to the next is independent of the utilization of the nodes

4. Work time homogeneity
   - The work time in the server is independent of its queue size

5. Arrival homogeneity
   - The arrival of new jobs into the system is independent of the number of jobs already in the system
Queuing Networks

- Queuing networks which apply to these constraints are called *separable* networks.
- Because each wait node can be analyzed independent of all others.
- Combining the performance results of all nodes gives the performance of the total queuing network.
Notations and Definitions

Fixed values:
- $T$ – observation time
- $K$ – number of nodes in the system

Measurable values:
- $A$ – observed arrivals
- $C$ – observed completions
- $A_k$ – number of arrivals for node $k$
- $C_k$ – number of completions for node $k$
- $B_k$ – busy time for node $k$
Basic definitions:

- $\lambda = \frac{A}{T}$ - arrival rate
- $X = \frac{C}{T}$ - throughput
- $\lambda_k = \frac{A_k}{T}$ - arrival rate for node $k$
- $X_k = \frac{C_k}{T}$ - throughput for node $k$
- $U_k = \frac{B_k}{T}$ - utilization for node $k$
- $S_k = \frac{B_k}{C_k}$ - average service requirement for node $k$ per job
Notations and Definitions (cont’d)

- \( V_k = \frac{C_k}{C} \) - average visit count for node \( k \) per job

- \( D_k = V_k S_k \) - average gross service demand

- \( D_k = \frac{B_k}{C} = \frac{U_k T}{C} \) - easy measurable

- \( D = \sum_k D_k \) - total service demand for the queuing network

Throughput Law:

\[
\frac{U_k}{V_k S_k} = \frac{B_k C_k C}{T B_k C_k} = \frac{C}{T} = X
\]
Notations and Definitions (cont’d)

- $N_k$ - average number of jobs for node $k$
- $N = \sum_k N_k$ - average number of jobs in the queuing network
- $W_k$ - average duration time for node $k$ per job
- $R_k = V_k W_k$ - average residence time for node $k$ per job
- $R = \sum_k R_k$ - total residence time per job in the queuing network
Foundational Operational Laws

- **Flow Balance Assumption**
  
  \( A = C \) or \( \lambda = X \)

- **Forced Flow Law**

  \[
  V_k = \frac{C_k}{C} \\
  C_k = V_k C \\
  X_k T = V_k X T \\
  X_k = V_k X
  \]

Each node has to perform an comparable amount of work in the fixed observation interval.
Forced Flow Law: Example

- Computer system with \( X = 2 \) jobs per second

- Each job needs to access a hard disk \( V_k = 8 \) times

- Therefore the hard disk needs to have a throughput of \( X_k = 16 \) accesses per second
Utilization Law

- Correlation between throughput and utilization for node $k$

\[ U_k = \frac{B_k}{T} \]

\[ U_k = \frac{C_k B_k}{T C_k} \]

\[ U_k = X_k S_k \]

\[ U_k = V_k X S_k = D_k X \]

- Example (modeling a hard disk)
  - Throughput of $X_k = 5$ jobs per second
  - Average service time per visit of $S_k = 0.11$ seconds
  - Therefore the disk has a utilization of $U_k = 0.55$, i.e. 55%.
Little’s Law

Arrivals and completions as function of time

\[ A(t) \]

\[ C(t) \]

Total duration time \( W \)
Little’s Law (cont’d)

- The vertical gap between $A(t)$ and $C(t)$ is the number of jobs in the system at time $t$
  - Therefore the average number of jobs in the system for time period $T$ is:
    $$N = \frac{W}{T}$$

- The horizontal gap between $A(t)$ and $C(t)$ is the duration time for a job
  - Therefore the average residence time of a job in the system is:
    $$R = \frac{W}{C}$$

Little’s Law:

$$N = \frac{W}{T} = \frac{C \cdot W}{T \cdot C} = XR$$
Little’s Law (cont’d)

- Little’s Law for nodes:
  - \( N_k \) corresponds to \( N \)
  - \( X_k \) corresponds to \( X \)
  - \( W_k \) corresponds to \( R \)

\[
\Rightarrow N_k = X_k W_k
\]

\[
N_k = X_k \frac{R_k}{V_k} = \frac{X_k}{V_k} R_k
\]

\[
N_k = X R_k \text{ (Forced Flow Law)}
\]

- \( U_k \) is equivalent to the average number of jobs in the server
  - Average number of waiting jobs (queue length): \( N_k - U_k \)
  - Average wait time: \( W_k - S_k \)
Little’s Law for Nodes: Example

(Hard disk again)

- $X_k = 5$ jobs per second
- $N_k = 2$ jobs in the node

$\Rightarrow W_k = 0.4s$

$S_k = 0.11s$

$U_k = 0.55$

$\Rightarrow W_k - S_k = 0.29s$

$\Rightarrow N_k - U_k = 1.45$ jobs are waiting on average
Summary of Foundational Laws for Operational Methods

- Forced Flow Law
  \[ X_k = V_k X \]

- Utilization Law
  \[ U_k = X_k S_k = D_k X \]

- Little’s Law
  \[ N = XR \]
  \[ N_k = X_k W_k = XR_k \]
Metrics of interest in queuing networks

- Throughput $X$
- Total residence time per job $R$
- Average number of jobs in the network $N$

Here only models with one class of jobs

Model parameters:

- Number of wait and delay nodes
- average gross service demand $D_k = V_k S_k$ to describe the workload
Open Models

- Needs also the average arrival rate $\lambda$
- Therefore:

$$X = \lambda$$

$$X_k = \lambda V_k$$

$$U_k = \lambda D_k$$

- There are no delay times in delay nodes (only service times)
  - $R_k = D_k$ for delay nodes
A job arriving at a wait node will be served after all other waiting jobs have left the node (FCFS)

There are $N_k$ jobs on average in the node

All nodes have a service demand time of $S_k$ on average

\[
W_k = S_k + S_k N_k \\
R_k = V_k W_k \\
R_k = V_k (S_k + S_k N_k) \\
R_k = D_k + D_k N_k
\]

\[
R_k = D_k + D_k X R_k \quad \text{(Little’s Law)} \\
R_k = D_k + D_k \lambda R_k \\
R_k = D_k + U_k R_k \\
R_k = \frac{D_k}{1 - U_k}
\]
Computing the average number of jobs in a node

Delay node:

\[ N_k = X R_k = \lambda D_k = U_k \]

Wait node:

\[ N_k = X R_k = \lambda \frac{D_k}{1 - U_k} = \frac{U_k}{1 - U_k} \]
Open Model: Example

- One processor with $D_{cpu} = 2.6s$
- Two disks with $D_{hdd1} = 3.4s$ and $D_{hdd2} = 1.3s$
- $\lambda = 0.2$ jobs per seconds

\[
\begin{align*}
U_{cpu} &= 0.52 \\
U_{hdd1} &= 0.68 \\
U_{hdd2} &= 0.26 \\
R_{cpu} &= 5.417s \\
R_{hdd1} &= 10.625s \\
R_{hdd2} &= 1.757s \\
N_k &= 1.038\text{ jobs} \\
N_{hdd1} &= 2.125\text{ jobs} \\
N_{hdd2} &= 0.351\text{ jobs}
\end{align*}
\]

- $X = 0.2$ jobs per seconds
- $R = 17.798s$
- $N = 3.56\text{ jobs}$