Performance Evaluation

"Statistics is the art of lying by means of figures" – Dr. Wilhelm Stekhel

Raj Jain, "*The Art of Computer Systems Performance Analysis*," Wiley and Sons Publishers, 1991.

What is Performance Evaluation?

- Performance of systems is the most important criterion while engineering systems as well as while buying systems.
- We want to get the highest performance for the lowest cost.
- Performance evaluation is the process to determine different aspects of performance of a system.
- Since some aspects can be subjective, and since performance evaluation does not have determined way to derive results: performance evaluation (PerfEv) is an art of computer science/engineering.
- Highly recommended book for your bookshelf: Raj Jain, "*The Art of Computer Systems Performance Analysis*," Wiley and Sons Publishers, 1991.

Why Performance Evaluation is an Important Tool in Networking

- Computer/telecommunication networks are extremely complex entities.
- The major part of research in telecommunications deals with determining whether a newly developed idea or component (protocol, algorithm, device) is better for the given usage scenario than the ones already existing.
- A novel idea/device/protocol/algorithm will only be accepted by the decision makers/managers/ community if it can be shown that it outperforms existing solutions (and to be able to show that so everybody can understand that – 2 reports.)

Techniques of PerfEv

- There are three basic techniques with which performance evaluation can be performed:

1. Measurement
2. Analytical Modeling
3. Simulations

Do not Cheat: The Ratio Game

- We want to compare two systems with the following measured performance:

System	Tput-1	Tput-2	Average
A	80	40	60
B	40	80	60

- How could the manufacturer of A benefit from these readings using the ratio game (B's performance is the unit)?

Common Mistakes

1. No goals (no general purpose model)
2. Biased goals (ours is better than theirs)
3. Unsystematic approach (arbitrary selection of system parameters)
4. Analysis w/o problem understanding (problem definition is half the problem solved)
5. Incorrect performance metrics (comparing apples with bananas)
6. Unrepresentative workload

Common Mistakes

7. Wrong evaluation technique
8. Overlooking important parameters
9. Ignoring significant factors (factors => parameters that are varied in the study)
10. Inappropriate number of experiments
11. Inappropriate level of detail (both ways)
12. No analysis (getting the results is not enough, neither is getting the right results => discussion)

Common Mistakes

13. Faulty analysis (e.g., too short simulation)
14. No sensitivity analysis (importance of various parameters)
15. Ignoring input errors (biased input)
16. Improper treatment of outliers
17. Assuming no change in the future
18. Ignoring variability
19. Analysis too complex

Common Mistakes

20. Improper presentation of results (has to be understandable quickly) – the number of analyses that helps decision makers.
21. Ignoring social aspects (social skills are often more important than technical)
22. Omitting assumptions and limitations.

How to Perform PerfEv?

1. State goals, and limitations of system.
2. List system services and possible outcomes.
3. Select metrics to be determined (e.g., throughput, BER). Speed, Reliability, Availability.
4. List parameters and select factors among them (and their resolution).
5. Select evaluation technique.
6. Select values for factors.
7. Design experiments
8. Analyse and interpret data.
9. Present results.

The Proper PerfEv. Method

- How easy it will be to convince people of your results.
- Analytical results are hard to “sell”, since most people do not understand them and thus are sceptical.
- Rules:
  1. Do not trust results from simulation until they are justified by analytics or measurements.
  2. Do not trust results from analytical studies until they are justified by simulations or measurements.
  3. Do not trust results from measurements until they are justified by analytics or simulations.
- Expert intuition: results should not be counter intuitive (especially for simulations)!

Common Performance Metrics

- Reaction time
- Response time (from start from finish)
- Turnaround time
- Throughput, capacity, usable capacity, efficiency
- Utilization (busy ratio)
- Reliability (prob. of errors)
- Availability (uptime, MTTF, MTBF).

Summarizing Data

Basic Probability...

- Independent events
- Random variable
- Cumulative distribution function {Fx(a)}
- Probability density function {f(x)- derivate}
- Probability mass function – (f(xi)=pi) discrete
- Mean (Expected) value
- Variance $\{\sigma^2 = E[(x-\mu)^2]\}$ – standard deviation
- Quantiles

Basic Probability...

- Median (0.5-quantile or 50-percentile)
- Mode - most likely value
- Normal Distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \qquad -\infty \le x \le \infty$$

  - The sum of n independent normal variates is a normal variate (normal processes remain normal)
  - The sum of a large number of independent variables from any distribution tends to have a normal distribution (CLT).

Confidence Interval for Mean

- (c_1,c_2) is called the confidence interval and $100(1-\alpha)$ is called the confidence level if $Pr(c_1 \le \mu \le c_2) = 1-\alpha$.
- The question can be, what is our confidence level that the mean value we have obtained by measurement is in a ±z% interval? Or, can we say that the result we have obtained is in a ±5% range with a confidence level of 95%?

Confidence Interval for Mean

- We are running a simulation that measures the mean access delay. How many different seeds do we have to run in order to claim that we are 95% sure that our result has less than 5% error?
- The standard error can be calculated by $z(\alpha)*s/\sqrt{n}$. (where s is the sample sets standard deviation, n is the size of the sample set and $z(\alpha)$ is the required normal quantile).

Confidence Interval for Mean

Confidence Level	$z(\alpha)$
80%	1.282
90%	1.645
95%	1.960
98%	2.326
99.9%	3.29

Simulation

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

Terminology

- State variables (define the system).
- Event (a change in the state).
- Continuous state/Discrete state models (state variables are cont/discrete).
- Deterministic/Probabilistic models.
- Static and dynamic models.
- Linear and non/linear models.
- Close and Open models (input is external).
- Stable and Unstable models (steady state).

Terminology

- Monte Carlo Simulation: simulation w/o a time axis, e.g., simulating the best strategy for Black Jack.
- Trace driven simulation: trace is a record of a real system that is used as an input.
  Discrete-event simulations: uses a discrete state model.

Discrete-Event Simulations

- Event scheduler: a linked list of what will happen. (e.g., schedule an event at time T, delay event by time dt)
- Simulation clock: can be unit time approach (generally not used) or it can be event-driven.
- System state variables: describe the state of the system.
- Event routines: functions handling the events scheduled.
- Input routines: to enter factors, set iterations and repetitions.

Discrete-Event Simulations

- Report generator.
- Initialization routines.
- Trace routines (debugging).
- Main program.

Validating/Verifying Simulations

- Validation and verification:
  - Invalid and unverified (assumptions invalid, simulation unverified).
  - Invalid and verified.
  - Valid and unverified.
  - Valid and verified.

Verification of Simulator

- Modular design (e.g., use of OOP).
- Antibugging (runtime checks in the program).
- Structured walk-through (explaining your model to someone else).
- Deterministic inputs (using inputs that have little randomness for verification).
- Running simplified simulations.
- Dumping traces (e.g., event logs) and going through them.
- Random variable/variate generations

Verification of Simulator

- Runtime graphic display (hardly used).
- Continuity tests (slightly changed input should not cause huge difference in the output).
- Degeneracy tests (running with lowest and highest possible parameters).
- Consistency test (2 sources with 30% load should have similar results than 4 sources with 15% load).
- Seed independence – cannot contradict for different seeds.

Validation of Simulations

- Validating:
  - Assumptions
  - Input parameters and their distributions
  - Output values and conclusions
- Tools:
  - Expert intuition
  - Real-system measurements
  - Theoretical results

Transient Removal

- Long runs
- Proper initialization
- Truncation, Initial data deletion, Moving average, batch means...
- Confidence intervalls!