

Overview of Performance Analysis

- **Introduction**
- **purpose of evaluation**
- **applications of performance evaluation**
- **performance evaluation techniques**
- **criteria for selecting an evaluation technique**
- **applicability of evaluation techniques**
- **steps for a performance evaluation study**
- **performance evaluation metrics**
- **capacity of a system**
- **performance evaluation study example**
- **common mistakes in performance evaluation**

- **Prerequisites:** Algorithms, data structure, probability, stochastic processes
- **Synopsis:** Evaluation and prediction of the behavior of computer systems and networks are integrated part of the design of these systems.
- This course will cover a set of techniques that are central to the modeling and performance evaluation of modern computer systems.
- These techniques are from the areas of
 - experimental design,
 - statistics (both parametric and non-parametric),
 - random number generation,
 - simulation,
 - queuing theory and queuing networks.


- This course will cover a set of techniques that are central to the modeling and performance evaluation of modern computer systems. These techniques are from the areas of experimental design, statistics (both parametric and non-parametric), random number generation, simulation, queuing theory and queuing networks
- General goals:
 - ☐ Determine certain performance measures for existing systems or for models of (existing or future) systems
 - ☐ Develop new analytical and methodological foundations, e.g. in queuing theory, simulation etc.
 - ☐ Find ways to apply theoretical approaches in creating and evaluating performance models

- Performance analysis is the study of the performance and behaviour of computer systems in order to make choices in the design, selection or procurement of these systems and their components that balances computer system performance with cost



Introduction

Computer system users, administrators, and designers are all interested in **performance evaluation** since the goal is to obtain or to provide the highest performance at the lowest cost.



A system could be any collection of H/W, S/W and firmware components; e.g., CPU, DB system, Network. Computer performance evaluation is of vital importance in the selection of computer systems, the design of applications and equipment, and analysis of existing systems.

Purpose of Evaluation

Three general purposes of performance evaluation:

- **Selection Evaluation** - system exists elsewhere
- **Performance Projection** - system does not yet exist
- **Performance Monitoring** - system in operation

Selection Evaluation



- **Evaluate plans to include performance as a major criterion in the decision to obtain a particular system from a vendor is the most frequent case**
- **to determine among the various alternatives which are available and suitable for a given application**
- **to choose according to some specified selection criteria**
- **at least one prototype of the proposed system must exist**

Performance Projection

- Orientated towards designing a new system
- to estimate the performance of a system that does not yet exist
- Secondary goal - projection of a given system on a new workload, i.e. modifying existing system in order to increase its performance or decrease its costs or both (**tuning therapy**)
- Upgrading of a system - replacement or addition of one or more hardware components

Performance Monitoring



- Usually performed for a substantial portion of the lifetime of an existing running system
- Performance monitoring is done:
 - to detect bottlenecks
 - to predict future capacity shortcomings
 - to determine most cost-effective way to upgrade the system
 - to overcome performance problems, and
 - to cope with increasing workload demands

Applications of Performance Evaluation

1. Procurement
2. System upgrade
3. Capacity planning : process of **predicting** when future load levels will **saturate** the system and of determining the most cost-effective way of delaying system saturation as much as possible.
4. System design

- Metric is something by which we judge a system.
 - This is what we observe, measure, evaluate.
- Parameter is what we change to see how it affects a metric.
- This difference in metrics and parameters is important to note in system evaluation

Performance Measures And Evaluation Techniques

1.1 Evaluation Metrics

A computer system, like any other engineering machine, can be measured and evaluated in terms of how well it meets the needs and expectations of its users.

It is desirable to evaluate the performance of a computer system because we want to make sure that it is suitable for its intended applications, and that it satisfies the given efficiency and reliability requirements.

We also want to operate the computer system near its optimal level of processing power under the given resource constraints.

All performance measures deal with three basic issues:

- How quickly a given task can be accomplished,**
- How well the system can deal with failures and other unusual situations,
and**
- How effectively the system uses the available resources.**

We can categorize the performance measures as follows.

Responsiveness:

These measures are intended to evaluate how quickly a given task can be accomplished by the system.

Possible measures are:

waiting time.

Processing time.

Conditional waiting time(waiting time for tasks requiring a specified amount of processing time),

Queue length. etc.

Usage Level:

These measures are intended to evaluate how well the various components of the system are being used.

Possible measures are:

Throughput and utilization of various resources.

Missionability:

These measures indicate if the system would remain continuously operational for the duration of a mission.

Possible measures are:

the distribution of the work accomplished during the mission time,

interval availability (Probability that the system will keep performing satisfactorily throughout the mission time),

life-time (time when the probability of unacceptable behavior increases beyond some threshold).

Dependability:

These measures indicate how reliable the system is over the long run.

Possible measures are:

number of failures/day.

MTTF(mean time to failure).

MTTR(mean time to repair).

Long-term availability, and cost of a failure.

These measures are useful when repairs are possible and failures are tolerable.

Productivity:

These measures indicate how effectively a user can get his or her work accomplished.

Possible measures are:

user friendliness.

Maintainability.

And understandability.

The relative importance of various measures

Because these measures are difficult to quantify, we shall not consider them.

The relative importance of various measures depends on the application involved.

In the following, we provide a broad classification of computer systems according to the application domains, indicating which measures are most relevant:

1. General purpose computing:

These systems are designed for general purpose problem solving.

Relevant measures are:

responsiveness, usage level. and productivity.

Dependability requirements are modest, especially for benign failures.

2. High availability:

Such systems are designed for transaction processing environments:

(bank, Airline. Or telephone databases. Switching systems. etc.).

The most important measures are responsiveness and dependability.

Both of these requirements are more severe than for general purpose computing systems,

moreover, any data corruption or destruction is unacceptable.

Productivity is also an important factor.

3. Real-time control:

Such systems must respond to both periodic and randomly occurring events within some(possibly hard) timing constraints.

They require high levels of responsiveness and dependability for most workloads and failure types and are therefore significantly over-designed.

Note that the utilization and throughput play little role in such systems.

4. Mission Oriented:

These systems require extremely high levels of reliability over a short period, called the mission time.

Little or no repair/tuning is possible during the mission.

Such systems include fly-by-wire airplanes, battlefield systems, and spacecrafts.

Responsiveness is also important, but usually not difficult to achieve.

Such systems may try to achieve high reliability during the short term at the expense of poor reliability beyond the mission period.

5. Long-life:

Systems like the ones used for unmanned spaceships need long life without provision for manual diagnostics and repairs.

Thus. In addition to being highly dependable, they should have considerable intelligence built in to do diagnostics and repair either automatically or by remote control from a ground station.

Responsiveness is important but not difficult to achieve.

1.2. Techniques of Performance Evaluation

There are three basic techniques for performance evaluation:

- (1) measurement.**
- (2) simulation. and**
- (3) analytic modeling.**

The latter two techniques can also be combined to get what is usually known as hybrid modeling.

In the following, we discuss these briefly and point out their comparative advantages and disadvantages.

1.2.1 Measurement

Measurement is the most fundamental technique and is needed even in analysis and simulation to calibrate the models.

Some measurements are best done in hardware, some in software, and some in a hybrid manner.

Measurement and Analysis

- Size, complexity and diversity of the Internet makes it very difficult to understand cause-effect relationships
- Measurement is necessary for understanding current system behavior and how new systems will behave
 - How, when, where, what do we measure?
- Measurement is meaningless without careful analysis
 - Analysis of data gathered from networks is quite different from work done in other disciplines
- Measurement/analysis enables models to be built which can be used to effectively develop and evaluate new techniques
 - Statistical models
 - Queuing models
 - Simulation models

Determining *What* to Measure

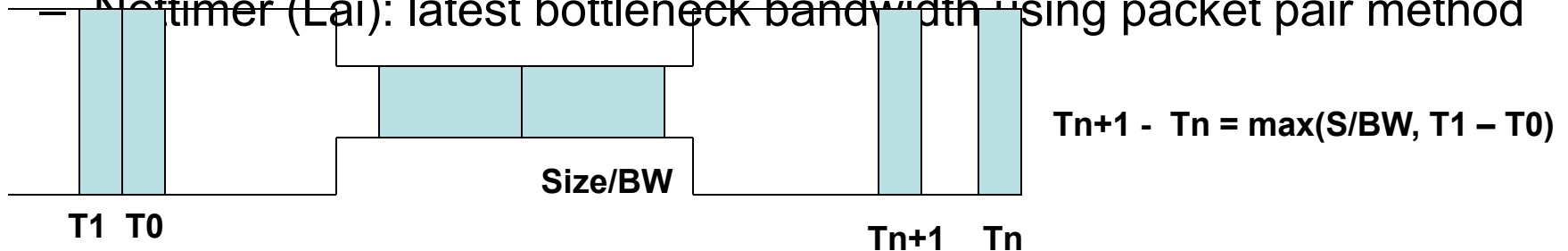
- Before any measurements can take place one must determine *what* to measure
- There are many commonly used network performance characteristics
 - Latency
 - Throughput
 - Response time
 - Arrival rate
 - Utilization
 - Bandwidth
 - Loss
 - Routing
 - Reliability

Measurement Introduction

- Internet measurement is done to either analyze/characterize network phenomena or to test new tools, protocols, systems, etc.
- Measuring Internet performance is easier said than done
 - What does “performance” mean?
 - Workload (what and where you’re measuring) selection is critical
 - Reproducibility is often essential
- Many tools have been developed to measure/monitor general characteristics of network performance
 - *traceroute* and *ping* are two of the most popular
 - These are examples of *active* measurement tools
 - Passive tools are the other major category
- Representative and reproducible workload generation will be a focus

Active Measurement Tools

- Send probe packet(s) into the network and measure a response
 - Ping: RTT and loss
 - Zing: one way Poisson probes
 - Traceroute: path and RTT
 - Nettimer (Lai): latest bottleneck bandwidth using packet pair method



- Pathchar: per-hop bandwidth, latency, loss measurement
 - Pchar, clink: open-source reimplementation of pathchar
- Problem: measurement timescales vary widely

Passive Measurement Tools

- Passive tools: Capture data as it passes by
 - Logging at application level
 - Packet capture applications (tcpdump) uses packet capture filter (bpf, libpcap)
 - Requires access to the wire
 - Can have many problems (adds, deletes, reordering)
 - Flow-based measurement tools
 - SNMP tools
 - Routing looking glass sites
- Problems
 - LOTS of data!
 - Privacy issues
 - Getting packet scoped in backbone of the network

Simulation



OVERVIEW:

- A simulation is any action that mimics reality.
- Chess simulates war.
- Monopoly simulates real estate investment.
- Any game is, in fact, a model of a reality.

Example:

- Computers simulate monopoly (that simulates)
- Computers simulate population growth/energy supplies.
- Computers simulate expansion of the Universe.
- Computers can simulate computers!!

1.2.2 Simulation Modeling

- Simulation involves constructing a model for the behavior of the system and driving it with an appropriate abstraction of the workload.
- Simulation is the realization of a model for a system in computer executable form.
 - That is, the model of the real-world system has been translated into a computer simulation language.
- The computer realization provides a vehicle to conduct experiments with the model
 - in order to gain insight into the behavior and makeup of the system or to evaluate alternatives

Simulation

Why Do Simulation

PROS AND CONS OF SIMULATIONS:

- The model can be as complicated as we wish (analytical models are limited by what we can write as an equation.)
- Must iterate a simulation many times - may take a long compute time.
- Simulations can be expensive to set up.
- Analytical modeling may be more compact and simple.

However, there are many important issues that must be considered in simulation:

1.It must be decided what not to simulate and at what level of detail. Simply duplicating the detailed behavior of the system is usually unnecessary and prohibitively expensive.

2.Simulation, like measurement. Generates much raw data. which must be analyzed using statistical techniques.

3.Similar to measurements. A careful experiment design is essential to keep the simulation cost down.

Both measurement and simulation involve careful experiment design, data gathering, and data analysis.

These steps could be tedious;

Simulation Process

The use of a digital computer to perform modeling and run experiments has been a popular technique for quite some time.

require discrete phases to be performed in order to realize their full potential

1. Determine that the problem requires simulation.
2. Formulate a model to solve the problem.
3. Formulate a simulation model of the problem.
4. Implement the model in a suitable language.
5. Design simulation experiments.
6. Validate the model.
7. Perform experiments.

The typical simulation model project will spend most of its time in phases 2, 3, and 4, because of the complexities associated with formulating the model and the conversion to simulation format and

Simulation Techniques

1. *Discrete models:*

- the real system's objects are typically referred to as entities.
- Entities carry with them attributes that describe them (i.e., their state description).
- Actions on these entities occur on boundary points or conditions. These conditions are referred to as events. Events such as arrivals, service standpoints, stop points, other event signaling, wait times, and so on are typical.
- For example, for the model of a self-service automatic teller machine, we need to define at a minimum the following entities and events:
 - Arrival events
 - Service events
 - Departure events
 - Collection events
 - Customer entities
 - Server entities

Simulation Techniques

2. *Continuous Modeling*

- Continuous simulations deal with the modeling of physical events (processes, behaviours, conditions) that can be described by some set of continuously changing dependent variables.
- These in turn are incorporated into differential, or difference, equations that describe the physical process.
- For example, we may wish to determine the rate of change of speed of a falling object shot from a catapult and its distance, R , from the catapult.
- Neglecting wind resistance, the equations for this are as follows. The velocity, v , at any time is found as:

$$v_x = v_0 \cos \theta$$

$$v_y = v_0 \sin \theta_0 - gt$$

Simulation

Pitfalls

COMMON MISTAKES:

Since modeling is not something to dive into lightly, what warnings are needed upfront?

Engineers inherently don't trust models. We tend to believe only things we can see. So you need to generate belief in your model among your "customers". Be very careful giving out answers to your model – don't do so before you believe the results yourself. Here are a number of issues you need to take into account:

- You need to get the **level of detail right!** It's very hard to know how much detail you need. There's a tendency among engineers for too much detail – after all, we're used to writing code. At the other extreme – too little detail can leave valuable mechanisms unaccounted for.
- **Unverified or invalid models.** You need to be constantly checking that your model matches reality. This is not easy. It requires finding things you can measure – such as the last version of the mechanism you're modeling.

Simulation

Pitfalls

COMMON MISTAKES:

- **The language of the simulation.** If the innards of the model require a lot of code (random number generators, event handlers, etc.) you can waste a great deal of time writing your own. Conversely, simple models don't need fancy simulation packages. More on this later.
- **Mechanical Issues.** The simulation didn't run long enough to be able to get the number of samples you need or to avoid end conditions. Your random number generator wasn't really random.

Simulation

Pitfalls

COMMON MISTAKES:

- **Software Engineering Issues.** In this management category falls things like:
 - a) Not knowing how long the model will take.
 - b) Lack of good software design. What starts out as a simple “hacked together” program becomes more and more complex and more and more unsupportable.
 - c) Not clearly defining the goals so you know when you’re done – someone always will ask a “what if” question that will keep you going forever.
 - d) Inadequate customer buy-in. Your customers don’t or won’t believe the outcome.
 - e) Inexplicable results.

Simulation Language

- As the use of simulation has increased, so has the development of new simulation languages.
- Simulation languages have been developed because of the unique needs of the modeling community to have system routines to keep track of time, maintain the state of the simulation, collect statistics, provide stimulus, and control interaction.
- All of these previously had to be done by each individual programmer.

GASP IV

- GASP IV was developed in the early 1970s as a general-purpose simulation language and is still in use with variations today. As such we use this as a basic model for most languages in existence today. GASP IV is a FORTRAN-based simulation language that provides routines and structure to support the writing of discrete events, continuous and combined discrete events, and continuous simulation models.
- **GPSS**
- General-Purpose Simulation System (GPSS) is a process-oriented simulation language for modeling discrete systems. It uses a block-structuring notation to build models. These provide a set of standard blocks that provides the control and operations for transactions (entities).

moreover, the final results obtained from the data analysis only characterize the system behavior for the range of input parameters covered.

1.2.3 Analytic Modeling

Analytic modeling involves constructing a mathematical model of the system behavior(at the desired level of detail) and solving it.

The main difficulty here is that the domain of tractable models is rather limited.

Thus, analytic modeling will fail if the objective is to study the behavior in great detail.

However. For an overall behavior characterization, analytic modeling is an excellent tool.

The major advantages of analytic modeling over the other two techniques are

(a) it generates good insight into the workings of the system that is valuable even if the model is too difficult to solve,

(b) simple analytic models can usually be solved easily, yet provide surprisingly accurate results, and

(c) results from analysis have better predictive value than those obtained from measurement or simulation.

1.2.4 Hybrid Modeling

A complex model may consist of several submodels, each representing certain aspect of the system.

Only some of these submodels may be analytically tractable, the others must be simulated.

We can take the hybrid approach. Which will proceed as follows:

1. Solve the analytic model assuming no fragmentation of memory and determine the distribution of memory holding time.
2. Simulate only memory allocation. Holding. And deallocation. And determine the average fraction of memory that could not be used because of fragmentation.
3. Recalibrate the analytic model of step 1 with reduced memory and solve it.
(It may be necessary to repeat these steps a few times to get convergence.)

1.3 Applications of Performance Evaluation

Performance modeling and evaluation may be needed for a variety of purposes;

However, the following needs stand out:

1. System design:

In designing a new system.

One typically starts out with certain performance/ reliability objectives and a basic system architecture.

And then decides how to choose various parameters to achieve the objectives.

This involves constructing a model of the system behavior at the appropriate level of detail, and evaluating it to choose the parameters.

At higher levels of design, simple analytic reasoning may be adequate to eliminate bad choice,

But simulation becomes an indispensable tool for making detailed design decisions and avoiding costly mistakes

2. System selection:

Here the problem is to select the “best” system from among a group of systems that are under consideration for reasons of cost, availability, compatibility, etc.

Although direct measurement is the ideal technique to use here.

There might be practical difficulties in doing so (e.g., not being able to use them under realistic workloads, or not having the system available locally).

Therefore, it may be necessary to make projections based on available data and some simple modeling.

3. System upgrade:

This involves replacing either the entire system or parts there of with a newer but compatible unit.

The compatibility and cost considerations may dictate the vendor,

So the only remaining problem is to choose quantity, speed, and the like.

Often, analytic modeling is adequate here;

however, in large systems involving complex interactions between subsystems. Simulation modeling may be essential.

Note that a direct experimentation would require installing the new unit first, and thus is not practical.

4. System tuning:

The purpose of tune up is to optimize the performance by appropriately changing the various resource management policies.

Some examples are process scheduling mechanism.

Context switching, buffer allocation schemes, cluster size for paging, and contiguity in file space allocation.

It is necessary to decide which parameters to consider changing and how to change them to get maximum potential benefit.

Direct experimentation is the simplest technique to use here, but may not be feasible in a production environment.

Since the tuning often involves changes to aspects that cannot be easily represented in analytic models, simulation is indispensable in this application.

5. System analysis:

Suppose that we find a system to be unacceptably sluggish.
The reason could be either inadequate hardware resources(CPU, memory, disk, etc.)

Or poor system management.

In the former case, we need system upgrade, and in the latter, a system tuneup.

Nevertheless, the first task is to determine which of the two cases applies.

This involves monitoring the system and examining the behavior of various resource management policies under different loading conditions.