Performance Evaluation of Computer Systems

By Ali Movaghar

Fall 2022

Grading Policy

- Programming Assignments: 20-30%
- Final Research Paper: 10%
- Exam(s): 60-70%

Teaching Assistants

- Iman Rahmati,

e-mail: imanrht@gmail.com

- Mahsa Ghaderan,

e-mail: mahsa.ghaderan@gmail.com

- Mohammad Hossein Khoshechin,

e-mail: mohammad.khoshechin76@gmail.com

Teaching Assistants

- Mehran Moeinijam,

e-mail: memoeinijam@gmail.com

- Seyed Ramtin Mousavi Gerayesh

e-mail: ramtin.mousavi13@gmail.com

- Amir Hosein Akbari Zarkesh,

e-mail: m_akbarizarkesh@yahoo.com

Textbooks

- **Main Texts:**
  - K. Kant, *Introduction to Computer System Performance Evaluation*, McGraw-Hill Inc., 1992.
  - M. Harchol-Balter, *Performance Modeling and Design of Computer Systems*, Cambridge University Press, 2013.
- **Secondary Texts:**
  - B.R. Haverkort, *Performance of computer Communication Systems*, John Wiley & Sons Ltd., 1998.
  - G. Bolch, S. Greiner, H. de Meer and K.S. Trivedi, *Queueing Networking and Markov Chains,* John Wiley & Sons Ltd., 1998.
  - D.W. Stroock, *An Introduction to Markov Processes,* Springer-Verlag, Berlin Heidelberg, 2005

Performance Measures

- **Responsiveness:** These measures are intended to evaluate how quickly a given task can be accomplished by the system. Possible measures are waiting 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.

Performance Measures (continued)

- **Missionability:** These measures indicate if the system would remain continuously operational for the duration of a 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), and the life-time (time when the probability of unacceptable behavior increases beyond some threshold).

Performance Measures (continued)

- **Dependability:** These measures indicate how reliable the system is over the long run. Possible measures are the number of failures per day, MTTF (mean time to failure), MTTR (mean time to repair), long-term availability, and the cost of a failure.

Performance Measures (continued)

- **Productivity:** These measures indicate how effectively a user can get his or her work accomplished. Possible measures are user friendliness, maintainability, and understandability.

Application Domains

- **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.

Application Domains (continued)

- **High availability:** Such systems are designed for transaction processing environments (banks, airlines, or telephone databases, switching systems, etc.). The most important measures are responsiveness and dependability. Both of these requirements are more sever than for general purpose computing systems. Productivity is also an important factor.

Application Domains (continued)

- **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 pay little role in such

Application Domains (continued)

- **Mission oriented:** These systems require 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 short term at the expense of poor reliability beyond mission period.

Application Domains (continued)

- **Long-life:** Systems like the ones used in 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.

Techniques for Performance Evaluation

- **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 hybrid manner.

Techniques for Performance Evaluation (continued)

- **Simulation Modeling:** Simulation involves constructing a model for the behavior of the system and driving it with an appropriate abstraction of the workload. The major advantage of simulation is its generality and flexibility; almost any behavior can be easily simulated.

Techniques for Performance Evaluation (continued)

- Both measurement and simulation involve careful experiment design, data gathering, and data analysis. These steps could be tedious; moreover, the final results obtained from the data analysis only characterize the system behavior for the range of input parameters covered. Although exploration can be used to obtain results for the nearby parameter values, it is not possible to ask "what if" questions for arbitrary values.

Techniques for Performance Evaluation (continued)

- **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.

Techniques for Performance Evaluation (continued)

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

  1) It generates good insight into the workings of the system that is valuable even if the model is too difficult to solve.
  2) Simple analytic models can usually be solved easily, yet provide surprisingly accurate results.
  3) Results from analysis have better predictive value than those obtained from measurement or simulation.

Techniques for Performance Evaluation (continued)

- **Hybrid Modeling:** A complex model may consist of several sub-models, each representing certain aspect of the system. Only some of these sub-models may be analytically tractable; the others must be simulated.

Applications of Performance Evaluation

- **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.

System Design (continued)

- At higher levels of design, simple analytic reasoning may be adequate to eliminate bad choices, but simulation becomes an indispensable tool for making detailed design decisions and avoiding costly mistakes.

Applications of Performance Evaluation (continued)

- **System selection:** Here the problem is to select the "best" system from among a group of system that are under consideration for reasons of cost, availability, compatibility, etc.

System Selection (continued)

- 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.

Applications of Performance Evaluation (continued)

- **System upgrade:** This involves replacing either the entire system or parts thereof 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.

System Upgrade (continued)

- 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.

Applications of Performance Evaluation (continued)

- **System tuning:** The purpose of tune-up is to optimize the performance by appropriately changing the various resource management policies. It is necessary to decide which parameters to consider changing and how to change them to get maximum potential benefit.

System Tuning (continued)

- 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.

Applications of Performance Evaluation (continued)

- **System analysis:** Suppose that we find a system to be unacceptably sluggish. The reason could be either inadequate hardware resources or poor system management. In the former case, we need system upgrade, and in the latter, a system tune-up. 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.

System Analysis (continued)

- Experimentation coupled with simple analytic reasoning is usually adequate to identify the trouble spots; however, in some cases, complex interactions may make a simulation study essential.

System Workload

- The workload of a system refers to a set of inputs generated by the environment in which the system is used, e.g., the inter-arrival times and service demands of incoming jobs, and are usually not under the control of the system designer/administrator. These inputs can be used for driving the real system (as in measurement) or its simulation model, and for determining distributions for analytic/simulation modeling.

Workload Characterization

- Workload characterization is one of the central issues in performance evaluation because it is not always clear what aspects of the workload are important, in how much detail the workload should be recorded, and how the workload should be represented and used.

Workload Model

- Workload characterization only builds a model of the real workload, since not every aspect of the real workload may be captured or is relevant.
- A workload model may be executable or non-executable. For example, recording the arrival instants and service durations of jobs creates an executable model, whereas only determining the distributions creates a non-executable model.

Workload Model (continued)

- An executable model need not be a record of inputs, it can also be a program that generates the inputs.
- Executable workloads are useful in direct measurements and trace-driven simulations, whereas non-executable workloads are useful for analytic modeling and distribution-driven simulations.

Benchmarking Computer Systems

- A benchmark of a system amounts to a set of published data about it.
- The benchmarks are primarily intended to provide an overall assessment of various types of a system on the market.
- Benchmarks are usually run by vendors or third parties for “typical” configurations and workloads, and not by the user interested in the selection process.

System Performance Evaluation Cooperative (SPEC)

- Recognizing the need for high quality standardized benchmarks and benchmark data on contemporary computer systems, a number of vendors have collectively established an organization called System Performance Evaluation Cooperative (SPEC).
- SPEC publishes a quarterly newsletter containing benchmark data on contemporary systems as they become available.

Benchmarking Traditional Computer Systems

- Two popular measures of the processing rate for conventional computer systems are MIPS (million instructions per second) and MFLOPS (million floating point instructions per second).
- Taken literally, these measures must necessarily be worthless, since the instruction formats, complexity, and execution times vary widely even for a single–machine type.

Benchmarking Traditional Computer Systems (continued)

- A reasonable approach must necessarily examine the running times of real programs written in a high-level language.
- We need to characterize the application domain by a set of “typical” programs.
- There are two ways to do this:

a) using application benchmarks.

b) using synthetic benchmarks.

Benchmarking Traditional Computer Systems (continued)

- In application benchmarks, we choose a small subset of real application programs that are representative of the application domain of interest.
- In synthetic benchmarks, we design some artificial programs that mimic a real program execution environment by using statistical data about real high-level language programs.

Synthetic Benchmarks: Some Examples

- Whetstone which is based upon the characteristics of Fortran programs doing extensive floating-point computation.
- Dhrystone which is written in C, and is designed to represent applications involving primarily integer arithmetic and string manipulation in a block-structured language.
- Nasa7 which consists of a set of seven kernels doing double-precision arithmetic in Fortran.

Application Benchmarks: Some Examples

- Linpack which solves a dense 100 X 100 linear system of equations using the Linpak library package.
- Spice which is a large analog circuit simulation package, mostly written in Fortran, which uses both integer and floating point arithmetic.
- gcc which is based on the GNU C compiler.
- li which is a lisp iterpreter written in C.

Reference Machine

- For historical reason, the VAX11/780 is considered as the reference machine. It is regarded to be a typical 1 MIPS (MFLOPS) machine.
- Thus, if an integer (floating-point) benchmark takes 80 seconds of CPU time on VAX11/780, and 4 seconds on machine A, we can claim that A is an 80/4 = 20 MIPS (MFLOPS) machine.

SPEC CPU Performance Benchmarks

- For CPU performance, SPEC has defined a suite of ten benchmarks, four of which (gcc, expresso, li, and eqnott) do primarily integer arithmetic, and other six (spice, doduc, nasa7, matrix, fpppp, and tomcatv) primarily floating point.
- The reference machine used is VAX11/780. The geometric mean of the integer benchmark results is known as SPECint, and those of others as SPECfp.
- The geometric mean of SPECint and SPECfp is known as SPECmark.

Geometric Mean Versus Arithmetic Mean

Let $t1$ and $t2$ denote the running times of two benchmarks on a test machine, and $r1$ and $r2$ denote the running times of two benchmarks on a reference machine. Then:

- $GM(r1/t1, r2/t2) = GM(r1, r2)/GM(t1, t2)$

but

- $AM(r1/t1, r2/t2) \neq AM(r1, r2)/AM(t1, t2)$

Benchmark Data for Some Selected Workstations

System	CPU/FPU	Mem	O/S	Mark	INT	FP
CDC CD4680	R6000A/6010	32 Mb	EP/IX 1.2.3	46.5	45.8	46.9
CDC DS5500	R3000/3010	32 Mb	Ultrix 4.1	21.5	21.9	21.1
HP 9000/400s	MC68040/int	16 Mb	HP UX 8.0	11.8	12.9	11.0
IBM RS6000/550	4164	64 Mb	Aix 3.1	54.3	34.5	73.5
Intel i860/40	I860/int	16 Mb	Unix860/4.0	26.7	19.9	32.5
MIPS RC6280	R6000/6010	32 Mb	Risc/OS4.52	46.5	45.0	47.6
SGI 4D/320S	R3000/3010	64 Mb	IRIX 3.3	19.5	22.6	17.6
Sun Sparcstn-2	CY7C601/T1	16 Mb	SunOS 4.1.1	21.2	20.7	21.5

Measurement

Measurement of a system concerns monitoring the real system. It can be broadly divided into into the following three classes:

- Hardware monitoring
- Software monitoring
- Hybrid monitoring

Hardware monitoring

- This technique employs additional monitoring hardware that is interfaced with the system under measurement in non-intrusive way.
- The main advantage of this technique is that the measurement does not interfere with the normal functioning of the monitored system and fast events can be captured.
- However, it is expensive and has difficulty in doing software-level measurements.

Software monitoring

- This technique uses some measurement code either embedded in the existing software or as a separate set of routines.
- The main advantage of this technique is its generality and flexibility.
- The disadvantages are that it may seriously interfere with the normal functioning of the system and cannot be used to capture fast occurring events.

Software monitoring (continued)

- This technique is most appropriate for obtaining user program and operating system related information, such as the time spent executing a particular routine, page fault frequency, and average number of processes in each possible state.

Hybrid monitoring

- This technique draws upon the advantages of both hardware and software monitoring.
- All relevant signals are collected under software control and sent to another machine for measurement and processing.

Hybrid monitoring (continued)

- The advantages are that it is flexible and that its domain of application overlaps those of both hardware and software monitoring.
- The disadvantages are that the synchronization requirements between the measuring and measured system may cause some interference, and it is expensive and cumbersome to obtain detailed program or O/S-level measurements.

Some Important Issues in Selecting an Appropriate Monitoring Technique

- Accessibility
- Event frequency
- Monitor Artifact
- Overhead
- Flexibility

Accessibility

- The hardware may be unaware of the software-level information and thus unable to obtain it. An example is the information regarding the allocation of various resources to a process.
- Similarly, the functions that are handled entirely in hardware such as cache management, and physical layer of networking may be inaccessible to software.