

EEC 686/785
Modeling & Performance Evaluation of
Computer Systems

Lecture 5

Wenbing Zhao
Department of Electrical and Computer Engineering
Cleveland State University
wenbing@ieee.org



Outline

- Review of lecture 4
- Capacity planning and benchmarking
- The art of data presentation



Monitors

- **Monitor:** a tool used to observe the activities on a system
- Usage:
 - To improve software performance. Find frequently used segments of the software
 - To measure resource utilizations and to find the performance bottleneck
 - To tune the system
 - To characterize the workload
 - To find model parameters, to validate models, and to develop inputs for models



Monitor Terminology

- **Event**
- **Trace**
- **Overhead**
- **Domain**
- **Input rate**
- **Resolution**
- **Input width**

Monitor Classification

- **By implementation**
 - Software monitor
 - Hardware monitor
 - Firmware monitor
 - Hybrid monitor
- **By trigger mechanism**
 - Event-driven: good for rare events
 - Timer-driven (sampling monitor): good for frequent events
- **By result display ability**
 - Online monitors: display the system state continuously
 - Batch monitors: collect the data that can be analyzed later

Program-Execution Monitor

- Software monitors designed to observe application software
- **Purpose**
 - **Tracing:** to find the execution path of a program
 - **Timing:** to find the time spent in various modules of the program
 - **Tuning:** to find the most frequent or most time-consuming sections of the code so that the system performance can be improved
 - **Assertion checking:** to verify the relationships assumed among the variables of a program
 - **Coverage analysis:** to determine the adequacy of a test run

Data in Accounting Logs

- **Resource usage**
 - CPU time
 - Elapsed time
 - Number and total size of disk I/O
 - Paging I/Os
 - Terminal I/Os
 - Network I/Os
- **Granularity**
 - System-wide resource usage
 - Resources used by each program
 - Resources used by each user session

Derived Quantities

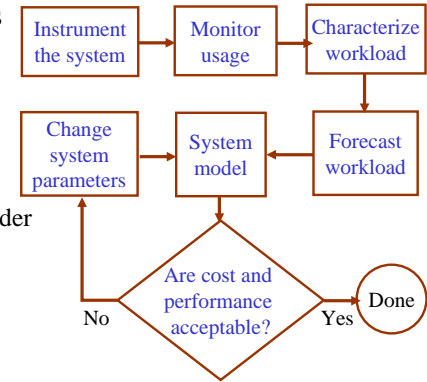
- **Per activation:** average resource consumption per activation of the program
- **Percentage of total:** resource consumed by all activations of a particular program, expressed as a percentage of the resources consumed by all activations of all programs
- **Per-second resource-consumption rates:** divide the resource consumption by the elapsed time
- **Per-CPU-second resource consumption:** divide the resource consumption by the CPU time consumed

Terminology

- **Capacity planning:** ensuring that adequate computer resources will be available to meet the future workload demands in a cost-effective manner while meeting the performance objectives
- **Capacity management:** ensuring that the currently available computing resources are used to provide the highest performance
- Capacity management is concerned with the present while capacity planning is concerned with the future
- **Benchmarking:** to compare the performance of two competing systems in an objective manner, benchmarks are run on these systems using automatic load drivers

Steps in Capacity Planning and Management

- For planning as well as management, the steps are basically the same:
 - Instrument the system
 - Monitor system usage
 - Characterize workload
 - Predict performance under different alternatives
 - Select the lowest cost, highest performance alternative



Problem in Capacity Planning

- There is no standard terminology
 - Some vendors use the term capacity management to include both capacity planning and tuning
 - Other use it to denote tuning only
- There is no standard definition of capacity
 - Maximum throughput, such as TPS, MIPS, bps
 - Maximum number of workload units, such as users, that the system can support

Problem in Capacity Planning

- There are a number of different capacities for the same system
 - Nominal, usable, knee capacity
 - Alternative terms: practical capacity (usable capacity), theoretical capacity (nominal capacity)
- There is no standard workload unit
 - Requires detailed characterization of the workload unit that varies from one environment to the next

Problem in Capacity Planning

- Forecasting future applications is difficult
 - Most of the forecasting is based on the assumption that the future trend will be similar to the past
- There is no uniformity among systems from different vendors
 - The same workload takes different amounts of resources on different systems
 - This requires developing a vendor-independent benchmark

Problem in Capacity Planning

- Model inputs cannot always be measured
 - Sometimes the input required for the model are not accurately measurable
 - E.g., Think time affected by coffee break
- Validating model projections is difficult
 - **Baseline validation:** requires using the current workload and configuration in the model and verifying that the model output matches the observed system performance
 - **Projection validation:** requires changing the workload and configuration and verifying that the model output matches the changed real system's performance

Problem in Capacity Planning

- Distributed environments are too complex to model
 - Interactions between different components are complex
 - The cost of individual components is not high enough to justify accurate modeling
- Performance is only a small part of the capacity planning problem
 - Key issue in capacity planning is cost, which include the cost of equipment, software, installation, maintenance, personnel, power, etc.
 - Performance modeling only addresses the sizing of equipment

Common Mistakes in Benchmarking

1. Only average behavior represented in test workload
2. Skewness of device demands ignored
3. Loading level controlled inappropriately
4. Caching effects ignored
5. Buffering sizes not appropriate
6. Inaccuracies due to sampling ignored

Common Mistakes in Benchmarking

7. Ignoring monitoring overhead
8. Not validating measurements
9. Not ensuring same initial conditions
10. Not measuring transient performance
11. Using device utilization for performance comparisons
12. Collecting too much but doing very little analysis

Benchmarking Games

- Differing configurations may be used to run the same workload on two systems
 - Different memory sizes, different number of disks, etc.
- The compilers may be wired to optimize the workload
- Test specifications may be written so that they are biased toward one machine
- A synchronized job sequence may be used
 - CPU-bound jobs and I/O-bound jobs are synchronized to achieve better performance

Benchmarking Games

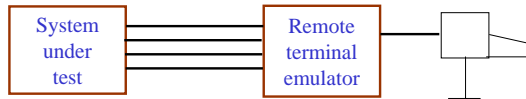
- The workload may be arbitrarily picked
 - Is the workload representative of real workload?
- Very small benchmarks may be used
 - Small workload can lead to high cache hits and hide I/O overhead
- Benchmarks may be manually translated to optimize the performance

Load Drivers

- In order to measure the performance of a computer system, it is necessary to have some means of putting loads on the system
- Load drivers also useful for
 - Component certification
 - System integration
 - Stress-load analysis
 - Regression testing

Load Drivers

- **Internal-driver method:** loading programs directly into the memory and executing
 - Problem 1: the effect of the terminal communication overhead is not visible
 - Problem 2: loading overhead may affect the system performance
- **Remote-terminal emulators (RTEs):** the computers that are used to put the load on the system



Remote-Terminal Emulation

Figure 9.3

- An RTE emulates
 - Terminals
 - Terminal communication equipments
 - Operators
 - Requests to be submitted to SUT

Remote-Terminal Emulation

- The user commands are read from a disk file called **script**. However, a script is system dependent
- Scenario: a description of workload in a manner that is independent of SUT and RTE

Figure 9.4

Components of an RTE

- Three phases
 - Preemulation
 - Emulation
 - postemulation

Figure 9.5

Limitations of RTEs

- The conditions for sending successive requests may not be realistic
- It may not be possible to vary the input in successive repetitions
- The think time model may not be realistic
- Modern user interfaces are not emulated
- Users are the shared resource
- Modern communication technology is not emulated
- Interface and definitions are different across RTEs
- Workstation emulation is not provided

Modern RTE

- **Modern RTE:** LoadRunner, commercially available from mercury.com (<http://www.mercury.com/us/products/performance-center/loadrunner/>)
- Info about LoadRunner: <http://www.wilsonmar.com/1loadrun.htm>

LoadRunner

- LoadRunner works by creating virtual users who take the place of real users operating client software, such as a Web browser, sending requests using the HTTP protocol to web servers
- Requests from many virtual user clients are generated by "Load Generators" in order to create a load on SUT
- The Controller controls load test runs based on "Scenarios" invoking compiled "Scripts" and associated "Run-time Settings"
- During runs, the status of each machine is monitored by the Controller
- At the end of each run, the Controller combines its monitoring logs with logs obtained from load generators, and makes them available to the "Analysis" program, which can then create run result reports and graphs

The Art of Data Presentation

- One of the important steps in every performance evaluation study is the presentation of final results
- The eventual aim of every performance analysis is to help in decision making
- Data presentation: prudent use of words, pictures, and graphs to explain the results and the analysis

Why Using Graphic Charts?

- A picture is worth a thousand words
 - A graphic chart saves readers' time and presents the same information more concisely
- It can also be used to interest the reader
- Most readers find it easier to look at the figures to quickly grasp the main points of the study and read the text only for details
- A graphic chart is also a good way to emphasize or clarify a point, to reinforce a conclusion, and to summarize the results of a study

Types of Variables

- The choice of a graphic chart is primarily determined by the type of variable to be displayed
- There are two types of variables
 - Qualitative variables
 - Quantitative variables

Types of Variables

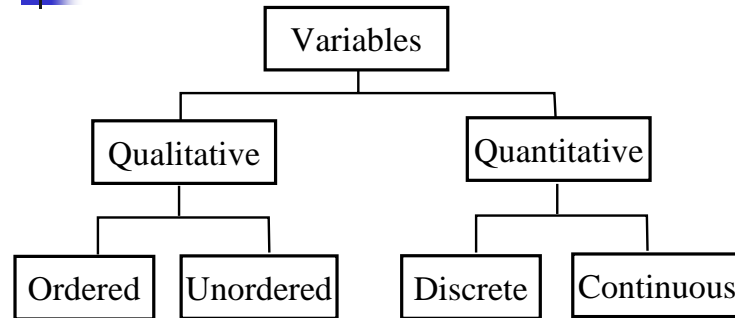
- **Qualitative variables**, or **categorical variables**: have states, levels, or categories that are defined by a set of mutually exclusive and exhaustive subclasses
 - The **subclasses** are usually expressed in words. They can be ordered or unordered
 - **Ordered** subclasses: a computer type with three subclasses – supercomputers, minicomputers, and microcomputers
 - **Unordered** subclasses: variable workload type with three subclasses – scientific workload, engineering workload, and educational workload

Types of Variables

- **Quantitative variables**: are those whose levels are expressed numerically. There are two types of quantitative variables: discrete and continuous
 - **Discrete quantitative variables**: all values of the variable can be counted and put into a one-to-one correspondence with some subset or all of the set of positive integers
 - Example: number of processors in a multiprocessor system
 - **Continuous quantitative variables**: can take uncountably infinite values. In many computer programming languages, they are called real variables
 - Example: the response time of a job on a system

Types of Variables

33



18 September 2005

EEEC686/785

Wenbing Zhao

Guidelines for Preparing Good Graphic Charts

34

- **Require minimum effort from the reader**
 - Example: there are two common ways of identifying different curves in a line chart: **legends** and **direct labeling**.
 - Since it requires a little more effort on the part of reader than direct labeling of the curves, **direct labeling is preferable** particularly if the number of curves is large

Figure 10.2

18 September 2005

EEEC686/785

Wenbing Zhao

Guidelines for Preparing Good Graphic Charts

35

- **Maximize information:** there should be sufficient information on the graph to make it self-sufficient
 - Use key words in place of symbols since the symbols require extra access to the reader's memory to make sense
 - The axes labels should be as informative as possible
 - ❖ "Daily CPU usage" is preferred over "CPU usage"
 - ❖ CPU time in seconds is more informative than "CPU time"

18 September 2005

EEEC686/785

Wenbing Zhao

Guidelines for Preparing Good Graphic Charts

36

- **Minimize ink:** present as much information as possible with as little ink as possible. Too much unnecessary info on a chart makes it cluttered and uninteresting
 - The chart that gives more info for the same data is preferable => (b) is preferred: it is more meaningful and informative than (a)

Figure 10.3

18 September 2005

EEEC686/785

Wenbing Zhao

Guidelines for Preparing Good Graphic Charts

37

- **Use commonly accepted practices:** present what people expect
 - Most people expect the origin to be (0,0)
 - Independent variable or cause to be plotted along the x-axis
 - The dependent variable or effect to be plotted along the y-axis
 - Scales to be linear, scales to increase left to right and bottom to top, and all scale divisions to be equal

18 September 2005

EEEC686/785

Wenbing Zhao

Guidelines for Preparing Good Graphic Charts

38

- **Avoid ambiguity:** show coordinate axes, scale division, and origin. Identify individual curves and bars. Do not plot multiple variables in the same chart

18 September 2005

EEEC686/785

Wenbing Zhao

Common Mistakes in Preparing Charts

39

- ***Presenting too many alternatives on a single chart***
 - In general, a line chart should be limited by 6 curves
 - A column chart or bar chart should be limited to 10 bars
 - A pie chart should be limited to 8 components
 - The number of cells and hence the number of bars, in a histogram should indicate that each cell has at least five data points

18 September 2005

EEEC686/785

Wenbing Zhao

Common Mistakes in Preparing Charts

40

- ***Presenting many y-variables on a single chart***
 - Reader has to associate the curves with the appropriate scales to get the message
 - Better to use 4 different graphs

Figure 10.4

18 September 2005

EEEC686/785

Wenbing Zhao

Common Mistakes in Preparing Charts

- *Using symbols in place of text*
 - The reader is left to flip through the report and search for what each symbol represents

Figure 10.5

Common Mistakes in Preparing Charts

- *Placing extraneous information on the chart*
 - The goal of each figure is to convey a particular message to the reader. Any info that detracts the reader from getting the message is extraneous and should be removed
 - For example, grid lines on a line chart should be there only if the reader is expected to read the values precisely

Common Mistakes in Preparing Charts

- *Select scale ranges improperly*
 - Most graphs are prepared automatically by programs that have several built-in rules to select the scales based on the min and max values seen in the data
 - In practice it is often necessary to manually override this automatic selection and specify the ranges to be shown

Common Mistakes in Preparing Charts

- *Using a line chart in place of a column chart*
 - The lines joining successive points on a line chart signify the fact that the intermediate values can be approximately interpolated
 - Example of misuse of a line chart: the performance in MIPS for various CPU types are plotted on the chart

Figure 10.6

Pictorial Games

- One can deceive as much with a picture as would require at least a thousand words
- Such pictorial games are more common in the trade press and in advertisements than in the technical literature

Techniques for Pictorial Games

- *Using nonzero origins to emphasize the difference*
 - By moving the origin and by appropriately scaling the graph, it is possible to magnify or reduce the perception of performance difference

Figure 10.7

MINE is much much better than YOURS MINE and YOURS are almost the same

Techniques for Pictorial Games

- **Three-quarter-high rule:** the right way to scale a graph is to choose scales such that the vertical height of the highest point is at least three-quarters of the horizontal offset of the right-most point
- The origin should be represented by (0,0) on the two axes unless there is a justifiable reason to do otherwise

The right way to draw the graph => **Figure 10.8**

Techniques for Pictorial Games

- *Using double-whammy graph for dramatization*
 - Two curves on the same graph can have twice as much impact as one
 - The example graph emphasizes that YOUR system performance goes down steeply as the number of users is increased

Your system is losing on both metrics => **Figure 10.9**

Techniques for Pictorial Games

■ *Plotting random quantities without showing confidence intervals*

- Most performance measurements result in a random quantification of the performance. If the measurement were to be repeated, the result would not be exactly the same
- It is necessary to draw the graph with confidential intervals if the variance is high

Figure 10.10

Techniques for Pictorial Games

■ *Pictograms scaled by height*

- To depict difference in performance is to draw appropriately scaled pictures of systems
- Correct way to scale a pictogram is by the area: if performance differs by 2, the height and width of MINE should be only 1.414 those of YOURS

Figure 10.11

Techniques for Pictorial Games

■ *Using inappropriate cell size in histograms*

- It is a difficult task to select appropriate cell (bucket) size
- Each cell should consist of at least 5 observations

Figure 10.12

Normal distribution?

Exponential distribution?

Techniques for Pictorial Games

■ *Using broken scales in column charts*

- An effect similar to that of nonzero origins discussed earlier is achieved in column charts and histograms by breaking the scale in the middle => amplify negligible performance difference

