

Computer Systems Performance Analysis and Benchmarking (37-235)

Analytic Modeling

Simulation

Measurements / Benchmarking

Lecture/Assignments/Projects:

Dipl. Inf. Ing. Christian Kurmann

Textbook:

Raj Jain, "The Art of Computer Systems Performance Analysis", 1991 Wiley & Sons, New York

Topic of Today:

- **Workload Characterization**
- **Monitors**
- **Logging**

Workload Characterization

To test multiple alternatives under the same condition, the workload should be *repeatable*. A real-user environment is generally not repeatable.

Procedure: study environment, characterize it and develop *workload model* that can be used repeatedly.

With workload model effects of changes in system or workload can be studied by simply changing model parameters.

Measured Workload Data consists of:

- services requested from SUT
- resource demands of users
- users (or better **workload unit**)
 - humans
 - programs
 - appliances

- workload characterization = characterizing a *typical* user or WL-component
- **workload components**
 - applications
 - sites
 - user sessions (logged)
- Measured quantities, requests, resource demands used to characterize the workload are parameters.
- **workload parameters**
 - transaction types
 - packet sizes
 - source/dest of packets
 - instructions
- workload parameters preferable over system parameters to characterize WL
- Include parameters with significant impact, exclude those with little impact.

Ways to specify workloads:

- Averaging
- Specifying dispersion (C.O.V)
- Single-parameter histogram
- Multi-parameter histogram
- Principle component analysis
- Markov models
- Clustering

Averages/Dispersion

Simplest method: Arithmetic mean

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

Alternatives: median, mode, geometric mean, harmonic mean.

Dispersion: Variability

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Standard Deviation s or Coefficient of Variation COV (std.dev/mean) are often more meaningful because of the same units.

Alternatives: range (min/max), percentiles, mean absolute deviation.

Case Study:Averages/C.O.Vs

Resource demands of various programs on six universities over 6 months were measured.

TABLE 6.1 Workload Characterization Using Average Values

Data	Average	Coefficient of Variation
CPU time (VAX-11/780)	2.19 seconds	40.23
Number of direct writes	8.20	53.59
Direct-write bytes	10.21 kbytes	82.41
Number of direct reads	22.64	25.65
Direct-read bytes	49.70 kbytes	21.01

TABLE 6.2 Characteristics of an Average Editing Session

Data	Average	Coefficient of Variation
CPU time (VAX-11/780)	2.57 seconds	3.54
Number of direct writes	19.74	4.33
Direct-write bytes	13.46 kbytes	3.87
Number of direct reads	37.77	3.73
Direct-read bytes	36.93 kbytes	3.16

Average demand by each progr. (Tab 6.1): high COV means that combination of all programs is bad.

Average demand for all editors(Tab 6.2): COV lower. Programs should be divided into several classes.

Histograms

- Histograms show relative frequencies of various values of a parameter.
- Complete parameter range is divided into several subranges (buckets) and observations that fall into a cell are counted.
- n buckets per histogram, m parameters per component, k components requires presenting $n*m*k$ values. Therefore only useful when variance high and mean cannot be used.
- Problem: No correlation among different parameters are shown. E.g. short jobs may lead to low number of Disk I/O but test workload may behave different.
- Multiparameter histograms for visualizing significant correlation.
- More than two parameter difficult to plot, even more detailed, rarely used.

Example Histograms:

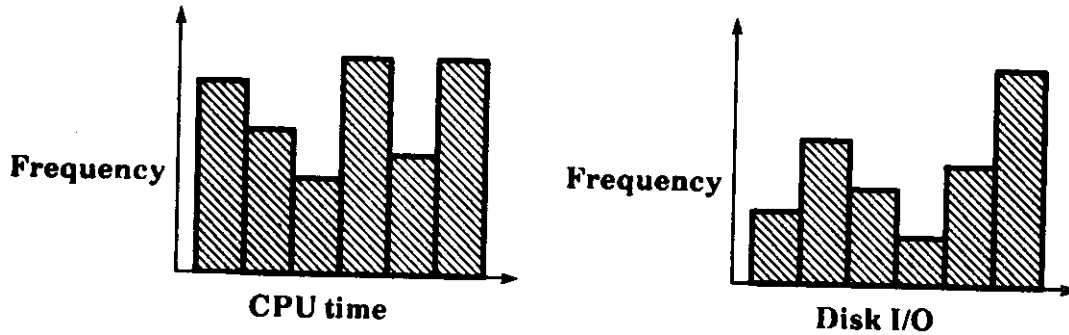


FIGURE 6.1 Single-parameter histograms of CPU time and disk I/O.

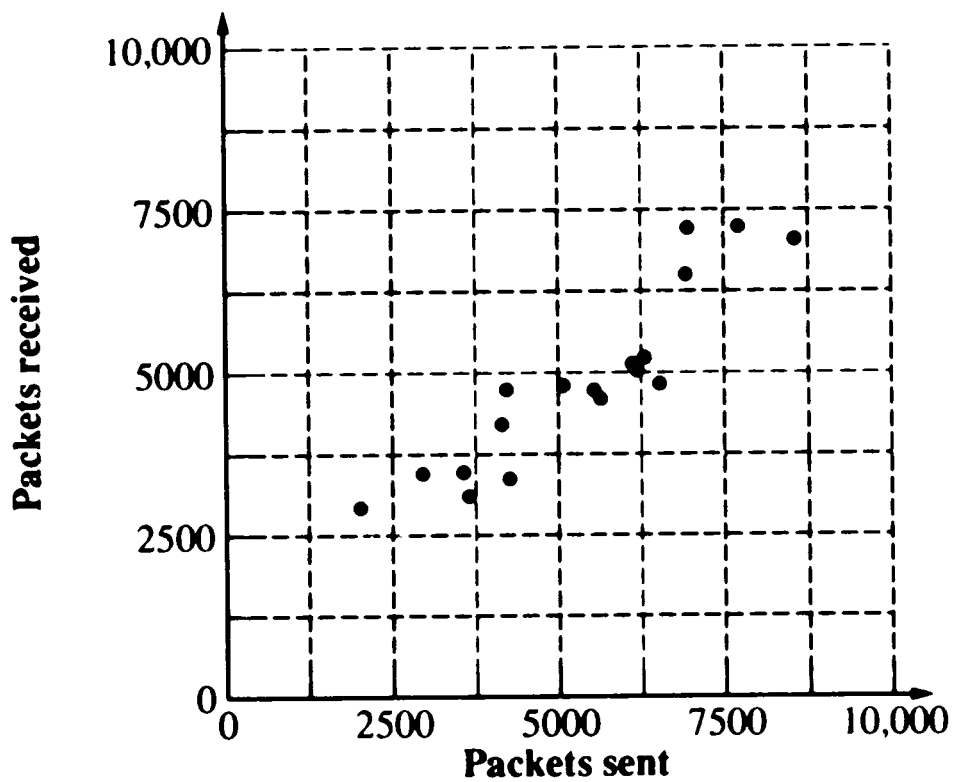


FIGURE 6.2 Two-parameter histogram.

Principle component analysis

Have many factors...but want a single one!

- Resource demands:
 - CPU, RAM, Disk...

want just:

- small, medium, large
- Determine some weights a_j for the j th parameter and compute weighted sum:

$$y = \sum_{j=1}^n a_j x_j$$

- This sum can be used to classify components into classes.
- Most performance analysis software ask user for main factors:
 - Do it manually by guess.
 - Better: Do it automatically by statistical criteria.

- Principle Component Analysis is used to determine the weights w_j such that y_j 's provide maximum discrimination.
- quantity y_j is called **principle factor**
- on a set of n parameters $\{x_1, x_2, \dots, x_n\}$ the principal component analysis produces a set of factors $\{y_1, y_2, \dots, y_n\}$ such that:
 - y 's are linear combinations of x 's:

$$y_i = \sum_{j=1}^n a_{ij}x_j$$

- y 's form an orthogonal set, inner product is therefore zero:

$$\langle y_i | y_j \rangle = \sum_k a_{ik}x_{kj} = 0$$

meaning y_i 's are uncorrelated.

- y 's form an ordered set such that y_1 explains the highest percentage of variance. Depending on the level of detail, only the first few factors can be taken to classify a WL-component.

Example

Number of packets sent and received, denoted by x_s and x_r , by various stations were measured. See scatter plot in Fig 6.2 histogram. There is a considerable correlation between the two variables. What is the principle factor?

TABLE 6.4 Data for Principal-Component Analysis Example 6.1

Observation No.	Variables		Normalized Variables		Principal Factors	
	x_s	x_r	x'_s	x'_r	y_1	y_2
1	7718	7258	1.359	1.717	2.175	-0.253
2	6958	7232	0.922	1.698	1.853	-0.549
3	8551	7062	1.837	1.575	2.413	0.186
4	6924	6526	0.903	1.186	1.477	-0.200
5	6298	5251	0.543	0.262	0.570	0.199
6	6120	5158	0.441	0.195	0.450	0.174
7	6184	5051	0.478	0.117	0.421	0.255
8	6527	4850	0.675	-0.029	0.457	0.497
9	5081	4825	-0.156	-0.047	-0.143	-0.077
10	4216	4762	-0.652	-0.092	-0.527	-0.396
11	5532	4750	0.103	-0.101	0.002	0.145
12	5638	4620	0.164	-0.195	-0.022	0.254
13	4147	4229	-0.692	-0.479	-0.828	-0.151
14	3562	3497	-1.028	-1.009	-1.441	-0.013
15	2955	3480	-1.377	-1.022	-1.696	-0.251
16	4261	3392	-0.627	-1.085	-1.211	0.324
17	3644	3120	-0.981	-1.283	-1.601	0.213
18	2020	2946	-1.914	-1.409	-2.349	-0.357
$\sum x$	96,336	88,009	0.000	0.000	0.000	0.000
$\sum x^2$	567,119,488	462,661,024	17.000	17.000	32.565	1.435
Mean	5352.0	4889.4	0.000	0.000	0.000	0.000
Standard Deviation	1741.0	1379.5	1.000	1.000	1.384	0.290

1. *Compute the mean and standard deviations of the variables:*

$$\bar{x}_s = \frac{1}{n} \sum_{i=1}^n x_{si} = \frac{96,336}{18} = 5352.0$$

$$\bar{x}_r = \frac{1}{n} \sum_{i=1}^n x_{ri} = \frac{88,009}{18} = 4889.4$$

$$\begin{aligned} s_{x_s}^2 &= \frac{1}{n-1} \sum_{i=1}^n (x_{si} - \bar{x}_s)^2 \\ &= \frac{1}{n-1} \left[\left(\sum_{i=1}^n x_{si}^2 \right) - n\bar{x}_s^2 \right] \\ &= \frac{567,119,488 - 18 \times 5352^2}{17} = 1741.0 \end{aligned}$$

Similarly,

$$s_{x_r}^2 = \frac{462,661,024 - 18 \times 4889.4^2}{17} = 1379.5$$

2. *Normalize the variables to zero mean and unit standard deviation. The normalized values x'_s and x'_r are given by*

$$x'_s = \frac{x_s - \bar{x}_s}{s_{x_s}} = \frac{x_s - 5352}{1741}$$

$$x'_r = \frac{x_r - \bar{x}_r}{s_{x_r}} = \frac{x_r - 4889}{1380}$$

The normalized values are shown in the fourth and fifth columns of Table 6.4.

3. *Compute the correlation among the variables:*

$$R_{x_s, x_r} = \frac{(1/n) \sum_{i=1}^n (x_{si} - \bar{x}_s)(x_{ri} - \bar{x}_r)}{s_{x_s} s_{x_r}} = 0.916$$

4. *Prepare the correlation matrix:*

$$C = \begin{bmatrix} 1.000 & 0.916 \\ 0.916 & 1.000 \end{bmatrix}$$

5. *Compute the eigenvalues of the correlation matrix. This is done by solving the characteristic equation. Using I to denote an identity matrix,*

$$|\lambda I - C| = \begin{vmatrix} \lambda - 1 & -0.916 \\ -0.916 & \lambda - 1 \end{vmatrix} = 0$$

or

$$(\lambda - 1)^2 - 0.916^2 = 0$$

The eigenvalues are 1.916 and 0.084.

6. *Compute the eigenvectors of the correlation matrix. The eigenvector \mathbf{q}_1 corresponding to $\lambda_1 = 1.916$ is defined by the following relationship:*

$$C\mathbf{q}_1 = \lambda_1\mathbf{q}_1$$

or

$$\begin{bmatrix} 1.000 & 0.916 \\ 0.916 & 1.000 \end{bmatrix} \times \begin{bmatrix} q_{11} \\ q_{21} \end{bmatrix} = 1.916 \begin{bmatrix} q_{11} \\ q_{21} \end{bmatrix}$$

or

$$q_{11} = q_{21}$$

Restricting the length of the eigenvector to 1, the following vector is the first eigenvector:

$$\mathbf{q}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 1 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Similarly, the second eigenvector is $\mathbf{q}_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 1 \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$

7. Obtain principal factors by multiplying the eigenvectors by the normalized vectors:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{x_s - 5352}{1741} \\ \frac{x_r - 4889}{1380} \end{bmatrix}$$

8. Compute the values of the principal factors. These are shown in the last two columns of Table 6.4.
9. Compute the sum and sum of squares of the principal factors. The sum must be zero. The sum of squares give the percentage of variation explained. In this case, the sums of squares are 32.565 and 1.435. Thus, the first factor explains $32.565 / (32.565 + 1.435)$, or 95.7%, of the variation. The second factor explains only 4.3% of the variation and can thus be ignored.
10. Plot the values of principal factors. The results are shown in Figure 6.3. Notice that most of the variation is along the first principal factor. The variation along the second factor is negligible.

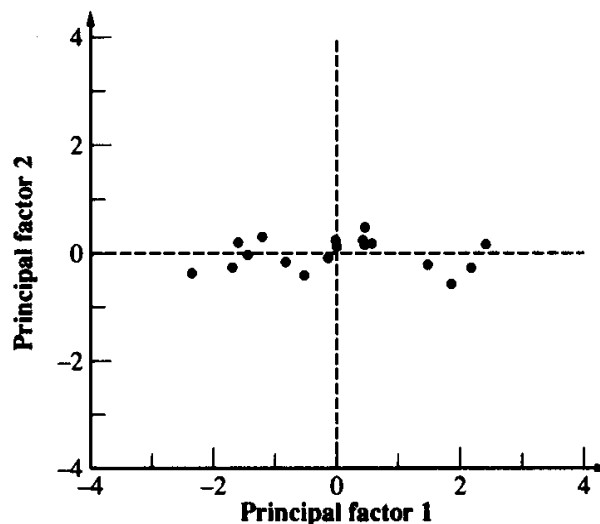


FIGURE 6.3 Packets sent and received data plotted along the principal-component axes.

Markov Models

Sometimes not only the number of service requests of each type but also their order is important.

When next request depends only on last request they follow a **Markov Model**.

Models used in Queuing Analysis

Transition matrix, which gives the probabilities of the next state given the current state, describe these models.

Limited memory behavior

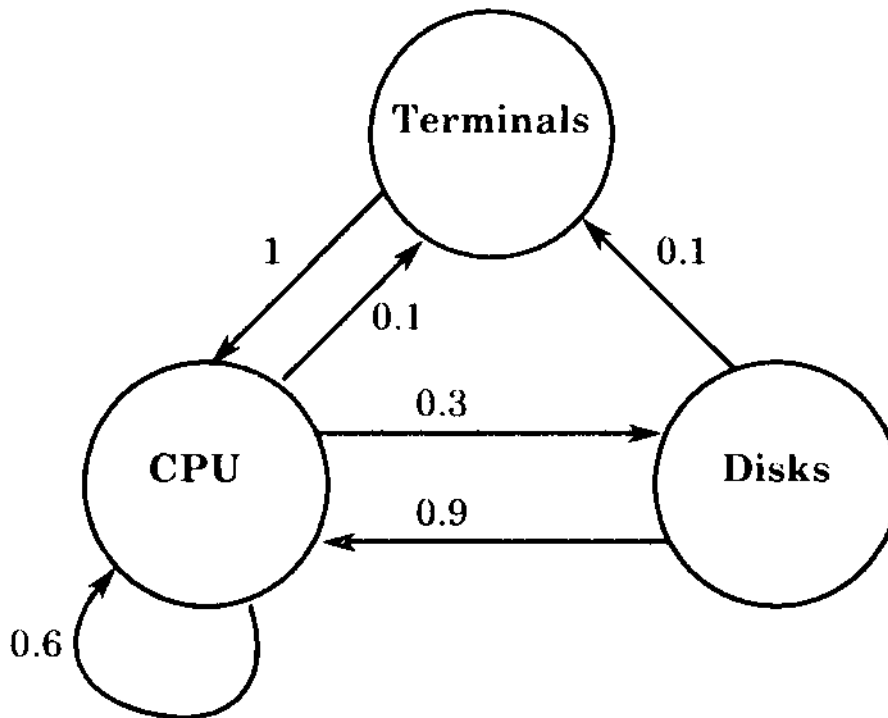
$P(\text{Transition to next state}) =$

$f(\text{current state}) \neq f(\text{all previous state})$

Example Markov Model

TABLE 6.5 Transition Probability Matrix

From/To	CPU	Disk	Terminal
CPU	0.6	0.3	0.1
Disk	0.9	0	0.1
Terminal	1	0	0



Monitors

A monitor is a tool used to observe the activities on a system, collect performance statistics, analyze the data and display results.

Monitors are used by:

- system programmer (optimization):
find frequently used segment in prog.
- systems manager (find bottleneck):
measure resource utilizations
- systems manager (tune system):
adjust system parameters
- systems analyst (charact. workload):
capacity planning, creating test WL
- systems analyst (find model params.):
validate models, develop inputs

Summary: Monitoring is the first and key step in performance measurements.

Monitor Terminology:

- Event = change in system state
(process context switch, packet arrival)
- Trace = log of events incl. time, type
- Overhead = perturbation by monitor
(CPU and storage demand, minimize!)
- Domain = activities observed
- Input rate = max frequency of event
(burst mode and sustained)
- Resolution = coarseness of information
(e.g. time resolution $>1\text{ms}$)
- Input width = number of bit sampled
(determines storage requirement)

Monitors Classifications

1. Dimension

- Event driven (activated by occurrence)
- Sampling (activated at fixed time interv.)
- Hybrid

2. Dimension

- On-line monitors (current system state)
- Batch (collect data to be analyzed later)

Methods:

- Software Monitors
- Hardware Monitors
- (Firmware Monitors)
- Hybrid Monitors

Software Monitors

Used to monitor OS and higher level software (network, DB...).

Generally they have lower input rates, lower resolution, higher overhead than hardware monitors. But they have higher input widths, higher recording capacities, are easier to develop and to modify.

Design Issues

- Activation Mechanism (Trap instruction / Trace mode/Timer interrupt)
- Buffer size / Number of buffers
- Buffer overflow
- Data compression/analysis on the fly
- On/Off switch / Language
- Priority
- Monitoring abnormal events

Hardware Monitors:

Consist of separate pieces of equipment, no system resources are consumed.

Thus they have generally lower overhead than software monitors, input rate is also higher, probability of introducing bugs into the system is low.

General-purpose hardware monitors consists of the following elements:

- Probes
- Counters
- Logic Elements
- Comparators
- Mapping Hardware
- Timer
- Tape/Disks

Software vs. Hardware Monitors

Choice not as difficult as it may appear.

Criterion	Hardware Monitor	Software Monitor
Domain	Difficult to monitor operating system events.	Difficult to monitor hardware events unless recognizable by an instruction.
Input rate	Sampling rates of 10^5 per second possible.	Sampling rate limited by the processor MIPS and overhead required.
Time resolution	10 nanoseconds is possible.	Generally 10 to 16 milliseconds.
Expertise	Requires intimate knowledge of hardware.	Requires intimate knowledge of software.
Recording capacity	Limited by memory and secondary storage. Not a problem currently.	Limited by overhead desired.
Input width	Can record several simultaneous events.	Cannot record several simultaneous events unless there are multiple processors.
Monitor overhead	None	Overhead depends upon the input rate and input width. Less than 5% adequate and more than 100% possible.
Portability	Generally portable.	Specific to an operating system.
Availability	Monitoring continues even during system malfunction or failure.	Cannot monitor during system crash.
Errors	Possible to connect the probes to wrong points.	Once debugged, errors are rare.
Cost	High	Medium

Distributed Systems Monitors

Distributed monitor functions divided in layers:

- **Observation:** Gathers raw data on individual component
- **Collection:** Collects data from various observers
- **Analysis:** Statistical summarization
- **Presentation:** Human user interface (generates reports, displays, alarms)
- **Interpretation:** Intelligent entity (human)
- **Console:** System control interface
- **Management:** Decision maker to set or change system parameters

Example Distributed Monitor:

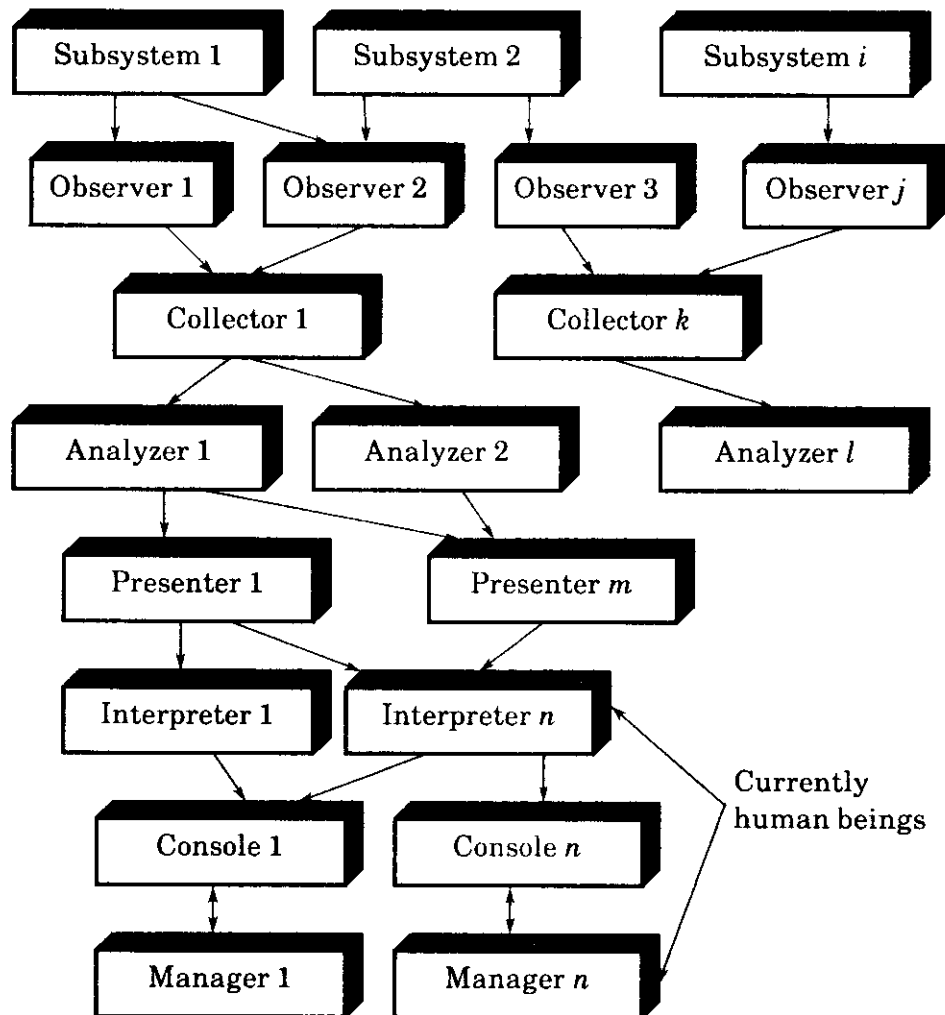


FIGURE 7.2 Components of a distributed-system monitor.