

COMPUTER PERFORMANCE EVALUATION

Dissertation submitted to the Jawaharlal Nehru University
in partial fulfilment of the requirements for the
award of the Degree of
MASTER OF PHILOSOPHY

KALPANA S. A.

**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY**

NEW DELHI - 110067

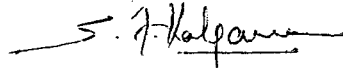
INDIA

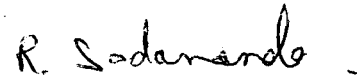
1984


Certificate

The research work embodied in this dissertation has been carried out in the School of Computer and Systems Science, Jawaharlal Nehru University, New Delhi - 110067, entitled "Computer Performance Evaluation".

This work is original and has not been submitted in part or full for any other degree or diploma of any other University.


Kalpana S.A.
Student


Dr. R. Sadananda
Supervisor


Dr. N.P. Mukherjee
Dean
School of Computer & Systems Sciences
Jawaharlal Nehru University
New Delhi - 110067.

ACKNOWLEDGEMENT

I acknowledge my guide Dr.R.Sadananda who has been remarkably encouraging all through.

I also wish to acknowledge the anonymous examiner who gave very valuable critical comments which enabled me to present my work in a better form. My heart felt thanks to him.

Lastly, my work has taken the shape as it is now due to the flawless and efficient typing of Mr.Chandrasekar.

Contents

1. Introduction . . 1
2. Objectives of Performance Evaluation . . 5
3. Mechanisms of evaluation . . 17
4. Quantitative evaluation of performance . . 23
5. Case Study : Computer Systems for educational use . . 31
6. Conclusion . . 42

Appendix I : Evaluation of the
HP1000 Architecture

Bibliography

1. INTRODUCTION

Every working system needs an evaluation of its performance for efficient running and satisfactory behaviour. For engineering and other technical systems there are some predetermined formulae and design criteria for evaluation. However, since the computer design trade-offs are changing at a rapid rate, evaluation methods are also gaining newer frameworks. There has been no distinct standard method designed as yet. An approach to a simple optimization technique for evaluation of the performance of a computer system has been the framework of this dissertation.

The characteristics chosen for evaluation vary widely with the kind of application the system has - Educational, Commercial, Scientific etc., and with the intention of the evaluation being conducted - study purpose, to buy a computer, to choose a system etc. This dissertation intends to give an overview of one of the techniques used for performance evaluation taking into account a few of the architectural features which are more important and interesting than most others.

In a similar manner, evaluation can be performed using the software features or any other attribute which might prove more important for some particular application. The characteristics of the specification prescribed for performance evaluation vary largely depending on the kind of system to be evaluated and on the perspective with which the

evaluator looks at it. Every system has some defined function or a set of functions to perform. Hence the first and foremost condition of performance is the satisfactory functioning of its defined function.

"Satisfactorily functioning" is technically termed as "correctness". Therefore correctness is the prerequisite for all other requirements. Verification of correctness is not subjective because for almost all systems, deciding whether they work correctly or not is comparatively easy. For example, a means of transport must move, a bulb must glow, a bridge must not collapse, and an amplifier must amplify an electric signal.

Once correctness is established, the next performance specification which must be satisfied, so that the user community might find the system acceptable, is the factor that how well the system performs its functions. Now arises the choice of performance requirements which is more subjective than laying down correctness specifications. However, performance requirement can be to a great extent, expressed quantitatively so that verification is not more complicated than the verification of correctness. For example, the maximum speed of a car with a load of 700 Kgs. is to be greater than 100 Kms/hr : the efficiency of an

electric transformer must be greater than 90% and its regulation must be less than 10% for a given load range. But, from an evaluator's perspective, what is more subjective is the relative importance given to the different aspects of performance; for example, the speed of a car may have a secondary importance with respect to the reliability, lifetime, fuel consumption, comfort, and ease of driving of the car.

To draw a strict margin between correctness and performance of a system is not really possible. Correctness can actually be looked upon as one of the important and essential aspects of performance. Hence in this dissertation, performance will indicate as to how well a system, assumed to perform correctly, works; only correctness is not enough to make a system acceptable to its users. Performance requirements also play an important role in the acceptance of a system by its users. For example, if a clock can move its hands at regular intervals of time, then it can be said to be performing its function correctly but unless it maintains the required accuracy it will not be acceptable to its users.

The way performance has been interpreted up to this point, it can be rightly said that performance is the technical

equivalent of the economic notion of value. That is, performance is the criterion which makes a system valuable to its users. But it is just one face of the coin in the real economic world. The other side is the cost. Hence, for each desirable system requirement, it is necessary to determine the cost of having it, the cost of not having it, the cost related to the dates at which it may be satisfied, and so on. However in this dissertation the cost aspects of the systems to be evaluated will not be considered because of the complications involved in relating directly the costs with the performance attributes in spite of the fact that performance and cost can't be separated and performance evaluation should always be accompanied by some form of cost evaluation.

Performance is actually a subjective concept. But there have been attempts to translate subjective definitions of performance into technical terms, and to quantify and hence to objectively evaluate performance. Performance evaluation can thus be regarded as a technical activity whose purpose primarily is the quantitative assessment of performance. A study of various aspects of the performance evaluation is collectively designated as Performance Evaluation Studies.

2. OBJECTIVES OF PERFORMANCE EVALUATION

Computers are hereby considered as engineering products. Performance evaluation of computers, though underdeveloped, is as essential as the other older branches of engineering. All computer systems are designed to perform certain functions related to the processing of information. The efficiency of these systems is a matter of technical, economical, and social importance. The different systems in a computer have different functions and nature. The different systems could be computer installations, computer systems, computer networks, system components, operating systems, programs, programming languages, and language translators. Their performance would be evaluated by their designers, manufacturers, managers, maintainers, and users. The objective of this dissertation is to evaluate computer systems in general and a few architectural features in particular.

A few definitions concerning system evaluation are as follows:

- a) Computer System : It is a collection of hardware components like central and input/output processors, memories, peripheral devices, interfaces, etc., and of software components like operating systems, compilers, assemblers, loaders etc.

- b) Resources : The different components of a system are called resources.
- c) System parameters : Each component has its own attributes which form the system parameters.
- d) Public software and public data bases : Software accessible to all the users of a system like compilers, editors, assemblers, debuggers, etc.
- e) Workload : A collection of inputs like programs, data, commands, etc., which are produced by the users is the workload.
- f) Load parameters : The attributes of the workload are called load parameters.
- g) Performance index : It is a descriptor which is used to represent a system's performance or some of its aspects.

PERFORMANCE INDICES:-

Performance being a subjective concept, the performance indices which one evaluator uses to assess a system may be different from the ones used by other evaluators. That is, if a set of computer systems in the same price range is considered, and a group of different evaluators is asked to rank them, then it is likely that each one of them will be ranked differently by different evaluators. This happens

because, each evaluator will use his own individual, subjective performance indices. Thus the ranking suggested by each person will show the value that person attaches to that system. This happens because of the subjectivity of performance aspects. So, subjectivity would be restricted to their choice and to the relative weights given to them. The weights proposed will differ for different categories of people like manufacturers, installation managers, company executives, field engineers, systems programmers, operators, application programmers and common users. Hence this subjectivity notion is changed to more objective, possibly quantifiable, technical Performance Indices.

But a problem arises when certain performance indices cannot be quantified. For example, the ease of use of a system, the structuredness of a program or of a language, and the power of an instruction set. Performance indices which can be quantitatively evaluated are the efficiency of a system, its speed in processing a given task or a set of tasks, and its promptness in responding to external stimuli.

The most popular classes of performance indices for computer systems are as follows:

1. PRODUCTIVITY :

DEFINITION : It is the volume of information processed by the system in a unit time.

EXAMPLES :

Throughput rate

Production rate

Capacity, i.e., maximum throughput rate

Instruction execution rate

Data processing rate

2. RESPONSIVENESS :

DEFINITION : It is the time between the presentation of an input to the system and the appearance of the corresponding output.

EXAMPLES :

Response Time

Turnaround Time

Reaction Time

3. PERCENT UTILIZATION :

DEFINITION : It is the ratio between the time a specified part of a system is used (or used for some specified purpose) during a given interval of time and the duration of that interval.

EXAMPLES :

Hardware Module (CPU, Memory, I/O channel, I/O device, Utilization).

Operating system module utilization

Data Base Utilization

DIMENSIONS :

Indices of productivity have the dimension volume \times time⁻¹, the indices of responsiveness have the dimension time, and the indices of utilization are dimensionless.

To measure the volume or quantity of information processed by a system, there is no standard unique way.

The different measures of volume used depend on the system and its workload, on the language used by the programmers to describe their algorithms to the machines, on the language of the machine itself, and on the way the system is organized.

DIFFERENCE BETWEEN PERFORMANCE ANALYSIS AND SYNTHESIS :

If the values of performance indices have to be determined for given values of installation parameters, then the task is called performance analysis. The converse, that is the realization of the parameter from given values of indices is called performance synthesis. Synthesis methods are used in

system design. The evaluation techniques used in this dissertation are essentially performance analysis techniques.

PHASES OF AN EVALUATION STUDY :

A concept close to performance evaluation study is the continuous monitoring of activities. The objective of continuous monitoring of activities is to keep the system's performance under observation in order to detect performance problems as soon as they arise.

The various phases of an evaluation study which are formed by grouping related activities together are as follows:

- PHASE I : The need for a study arises
- PHASE II : The objectives of the study are formulated.
- PHASE III : A plan of the study is prepared.
- PHASE IV : The plan is implemented.
- PHASE V : The results are interpreted.

Every phase listed above will include a few or all of the activities aimed at gathering information about the system and its installation; reading system manuals, examining logic diagrams, flowcharts and program listings; questioning operators and other system installation staff members; interviewing users; measuring workload parameters; collecting data about system usage; interpreting accounting data; and so on.

CLASSIFICATION OF EVALUATION STUDIES :

Like all other studies, evaluation studies may also be classified along different dimensions depending upon their objectives. But, the most conventional classification is the one mentioned below dividing it into three categories :

- a) Selection Studies
- b) Improvement Studies
- c) Design Studies

SELECTION STUDIES : A need for this kind of study arises when a new installation has to be configured and procured. These problems include :

1. **Selection of the processing mode :** It is concerned with the installation being set up and with the processing mode that will be most useful to the user community. The main modes are batch, interactive, open-shop or closed shop, direct access, and real-time. Selecting the processing mode gives the installation designers information regarding effectively restricting the field of acceptable hardware and software solutions.
2. **Vendor Selection :** The main problem that arises during the design of a new installation and the procurement of a new system in place of an existing computer installation is that

of vendor selection. Many other factors like the economical and technical constraints restrict this problem to a smaller perspective. But, all the same, it still exists because a choice has to be made out of a number of alternatives whose ability to meet the performance requirements of the installation vary. Hence a comparison has to be made. While making this kind of competitive selection, various alternatives are proposed by different vendors who in turn offer one or more configurations based on the design of their hardware and software components. So, a selection made under such circumstances is termed as vendor selection.

3. Installation selection : To satisfy a given information processing need, there can be several available installations. Choosing the most suitable one for the required application is the Installation selection problem. It could be selecting among the different computing facilities in an organization, by a department, a group or an individual programmer to satisfy his needs. It could also be the selection of a service bureau which is a company or an external installation which sells computer time for the processing of a given workload. In some ways this problem is similar to the vendor selection. However, they differ in their economic aspect.

4. System Component Selection : This class deals with the upgrading or expanding problems of a given system, thus improving upon the current system. That is, some components must be added to the already existing systems, running in an existing installation. The components to be selected can be software modules like operating systems, accounting packages, software measurement tools, compilers, data-base systems, etc., and hardware modules like primary-memory boxes, central and I/O processors, disc drives, tape drives, multiplexers, and so on.

5. Application Program Selection : In this class the selection is done precisely by the particular user and not by the installation's management, some examples of problems of this kind are :

 a) The selection of one among several languages offered by an installation for the coding of a given program.

 b) The selection of one among several packages existing in an installation for the execution of a given task.

 c) The selection of one among several packages existing in the market for the satisfaction of a certain information processing requirement.

2. **IMPROVEMENT STUDIES :** This kind of study comes into picture when there is an installation already existing and some modification has to be made to improve its performance or decrease its cost, or both. Another problem encountered in improvement studies is that the system has to be tuned; that is, its parameters have to be adjusted so that it could suit some new environment. This, in turn, results in the upgrading of a system by either adding one or more components or by optimizing the already existing resources. As examples of system upgrading are - increasing the size of the primary memory, adding one or more CPU's or replacing an old CPU by a faster one, rearrangement of information within one of the storage devices, or among several of them, and the modification of the connections between I/O devices and I/O processors. Thus the outcome of improvement studies are that : the same workload can be processed in a shorter time, and in this way reducing the number of operations staff involved; better performance attracts more customers; the procurement of larger, more expensive system may be postponed or avoided because of the increased ability of the current system to cope with an expanded workload; and the addition of a hardware component to the configuration may be made unnecessary by the changes suggested by an improvement study. Modification also means to determine as to which components seem superfluous and can be

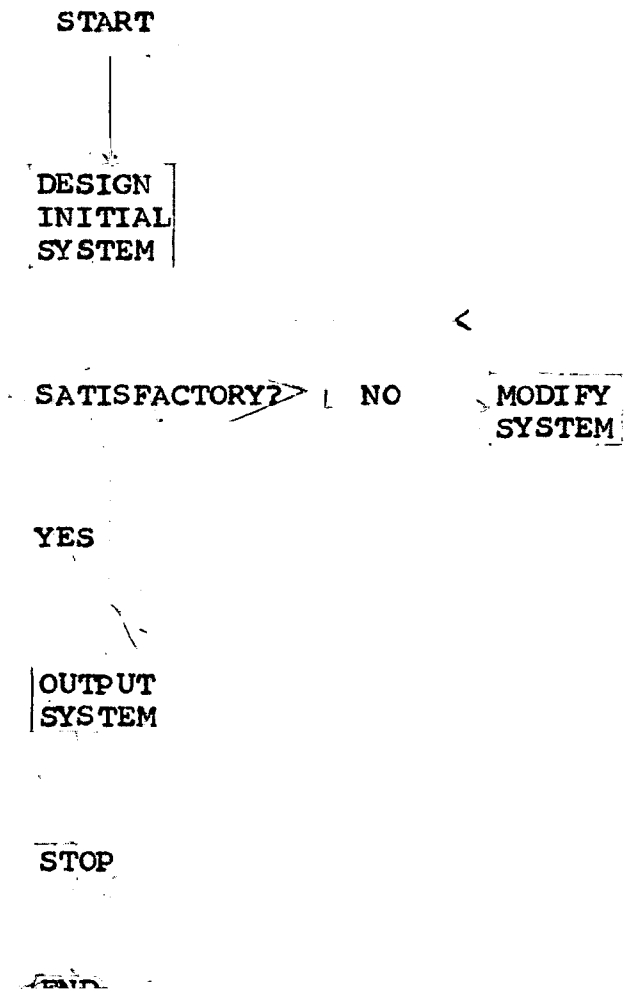
removed so that the system's performance will not change appreciably. An improvement study may result in a substantial increase in performance with relatively little effort.

3. **DESIGN STUDIES** : They are concerned with the performance evaluation of the design of computers, computer components, operating systems, programs, and languages. Unlike the other two classes where the study is undertaken by the users, in this class the study is done by the manufacturers. This is done to determine how a computer system capable of satisfactorily handling certain information processing tasks can be constructed. Here a lot of creativity is needed. The pre-existing components should be properly made use of and combined in new fashions to meet new requirements.

The design problem is stated as follows :

Produce an implementable description of a system which satisfies the given design specifications. The specifications could be - functional specification, performance specification, or cost specification.

The iterative design methodology is shown by the following flow-chart.



3. MECHANISMS OF EVALUATION

The attributes considered for evaluating a system will vary widely with the purpose and perspective of evaluation. To start with, the techniques of measurement, evaluation and benchmarking are considered in general in this chapter.

Two broad classes into which measurement techniques can be grouped are hardware techniques and software techniques. Hardware techniques are further classified into standard features which can be used for measurement and special hardware-instrumentation techniques. Software techniques can be classified as simulation and analytical models or measurement processes within the system.

HARDWARE MEASUREMENT

In the early stages of system development standard hardware features were incorporated. For example, the IBM-650 allowed a person to set an address breakpoint by switches at the console. The machine would run until this address at which the breakpoint was set was reached and then it would stop. At this stage the programmer could examine

the registers or dump memory for further analysis. In later stages of system development like the IBM-704 a mechanism called the trapping transfer mode was introduced. In this mode whenever a transfer instruction was reached the computer would interrupt itself and control would be transferred to some fixed location where the user program recorded the event. Even these days on current machines similar features exist but are operable only from the maintenance panel. These features were initially incorporated only to simplify program debugging. They did not prove very useful for system measurement.

A representative of a hardware measuring device is IBM's Systems Performance Activity Recorder (SPAR). It consists of 256 high-impedent probes. They resemble the probes used on an oscilloscope. They can be connected to various circuit points to obtain logic pulses or levels; several counters; clocks; basic logic elements such as AND gates, OR gates, complement gates, flip-flops and comparators,

a plug board; and the ability to store the counter contents on magnetic tape.

The probes can be connected to address lines, the instruction counter, mode flip-flops, or other control and information points. Then an event measure is created by routing the probe signals, or more often the probe signals in combination with a clock signal, through the logic elements by plugboard programming. The signal which represents the event is plugged into a counter. These counters can be dumped on to a tape at fixed intervals of time or dumped on to a tape under event control. Some systems like the SNUPER at UCLA have adopted the above-mentioned approach one step further. In this the event signals and counters are connected directly to a general purpose digital computer. The output of this computer can be printed.

The advantage of this kind of hardware measurement technique are that it (1) does not interfere with the operation of the system, (2) allows very fine measurements at the microsecond or submicrosecond level, as well as coarser measurements, such as percent CPU utilization, and

(3) allows access to all parts of the system.

The obvious disadvantages of the hardware measurement technique are that :-

1. The experiments must be carefully planned with a knowledge of the hardware.
2. Only a limited number of experiments can be undertaken at one time because the number of probes, counters, and logic elements is limited.
3. The plugboard program must be carefully debugged.
4. A person who knows the hardware is required to perform the measurement, and
5. A considerable amount of time is required to set up each experiment.

SOFTWARE TECHNIQUES

SIMULATION : It is a very common software technique for measurement. The levels of detail at which simulation models can be produced depends on the problems under consideration. Hence it must be used with discretion.

The types of data that are useful in simulation are :-

1. Distribution of user thinking time at the console for various types of functions.
2. Distribution of process memory requirements.
3. Distribution of processor time required to service a request.
4. Distribution of time between blocking for terminal I/O or other I/O, and
5. Distribution of shared memory requirements. The simulator can then pick parameters from these distributions for processes and use these data as an input to the model. The output of the model will be information about response time, equipment utilization, etc.

ANALYTIC MODELS : This involves mathematics like determining the algebraic relationships of the variables under study or some sophisticated techniques for studying random processes. Due to the lack of advanced mathematical knowledge and due to number of simplifying assumptions required, analytical modelling techniques have not been very successful.

INTERNAL SYSTEM MEASUREMENT : They fall into three main categories.

1. Event counting which simply counts the number of times an event occurs in a given time.
2. Trace techniques which record data about events in the sequence in which they occur.
3. Sampling techniques which periodically interrupt the system and record the status of registers, calls, tables, other data structures, hardware units, and so on.

BENCHMARKING : It arises because of the need to derive suitable yardsticks by which valid comparisons between various systems can be made. Benchmarking represents a traditional approach to system evaluation. In this a suitable program whose functioning furnishes the required information about a particular attribute is designed. This program is taken as the benchmark. The results produced by the same benchmark on various systems can be used for comparison.

4. QUANTITATIVE EVALUATION OF PERFORMANCE

This chapter involves a procedure for an evaluation method which can be used (a) to compare alternative computer system designs with respect to specific problem requirement, (b) to determine the effect of change in key system parameters and (c) in an iterative design improvement process. Hence this model can characterize current system performance.

THE APPROACH

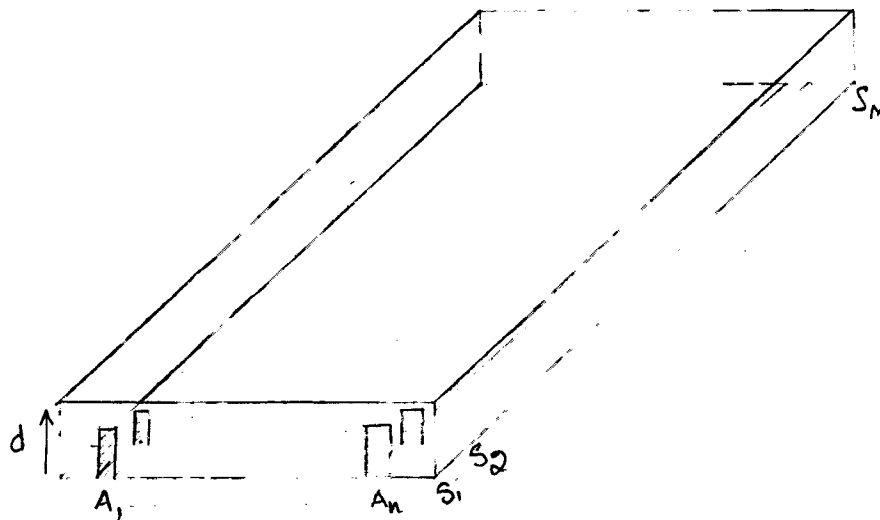
The framework involves the following components.

1. A set of attributes or descriptors.
2. A measure of how much of each of these attributes is required for a particular problem.
3. An indication of the extent to which a structure possesses each of these attributes for that problem, and
4. An indication of the **relative** importance of each attribute to the problem being addressed.

These elements are combined to provide a ranking from which the most suitable candidate can be selected.

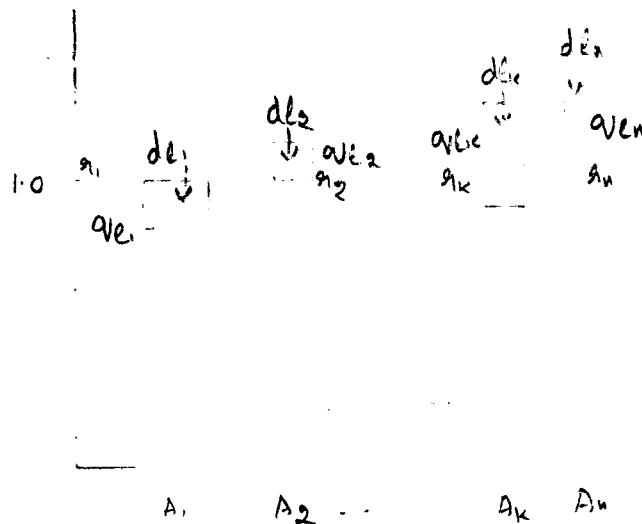
Let $A = A_1, A_2, \dots, A_n$ be the set of attributes used to describe problem requirements and system characteristics. These attributes should be obtained independently of any specific problem, and all attributes in the set should be used to assess the merits of each candidate system.

Let $S = S_1, S_2, \dots, S_n$ represent the set of candidate systems considered for a particular problem. In case only one system is evaluated for its performance then the set S will consist of just one element. Each of these systems will be evaluated in terms of A .



The figure shows how the capabilities of individual computer systems can be compared in terms of the elements of the attribute set A .

Let $R = \{r_1, r_2, \dots, r_n\}$ represent problem requirements such that r_i represents the amount of A_i required for a given problem, and let the $N \times n$ matrix Q describe the attributes of all the candidate systems such that q_{ji} represents the extent to which systems S_j , $1 \leq j \leq N$ possesses attribute A_i , $1 \leq i \leq n$. The weighted differences between R and Q will finally give the figure of merit that will eventually result in the desired ranking. The importance of each attribute has to be quantitatively expressed. The order of the importance of each has to be chosen, i.e., which attribute has the highest importance, which is the next and so on. To accomplish this, the chosen attribute set A is rearranged in descending order of importance.



The width of each rectangle gives the weight or importance of each attribute. Let the set representing the weights be

$$W = \{W_1, W_2, \dots, W_n\}$$

Next, the weighted attribute d_{li} has to be obtained. The weight of each attribute is multiplied by the amount of attribute by which the system exceeds the given requirement.

For a system S_l the weighted difference D_l is

$$D_l = \sum_{i=1}^n d_{li} = \sum_{i=1}^n (\alpha_{li} - 1) w_i$$

$$\text{where } \alpha_{li} = v_{li} / r_{li}$$

Thus the set $D = \{D_1, D_2, \dots, D_N\}$ represents the set of weighted difference, for N systems.

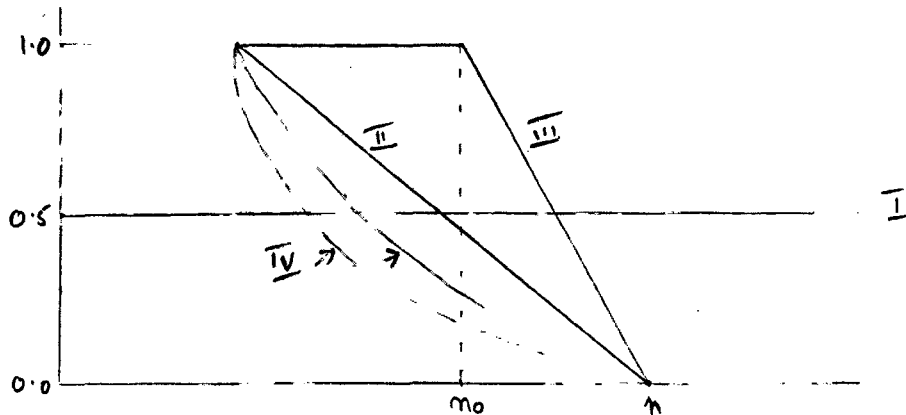
If S_k represents the most suitable system, then

$$D_k = \text{Max. } \{D_1, D_2, \dots, D_N\}, \quad 1 \leq k \leq N$$

If only one system is considered then its percentage efficiency will be given as

$$\eta = \frac{D_l}{m} \times 100 \%$$

Now, the set of weights W has to be determined and assigned to attributes



The figure shows the four ways in which the weights can be assigned.

Class I : All attributes are assigned the same weights.

$$W_i = 1$$

Class II : Attributes have a linear relationship such that A_1 has a weight 1.0 and A_n has a weight zero. The intermediate weights will depend on the slope

$$W_i = 1 - \left(\frac{i-1}{n-1} \right),$$

$$1 \leq i \leq n$$

Class III : The first n_0 attributes have a weight 1.0 and the remaining $(n-n_0)$ attributes display class II characteristics.

$$W_i = \begin{cases} 1, & 1 \leq i \leq n_0 \leq n \\ 1 - \frac{1-n_0}{n-n_0}, & n_0 \leq i \leq n \end{cases}$$

Class IV : Exponential curves. Highest importance is attached to the first attribute and the remaining attributes become relatively unimportant depending on the constant in the exponent (between 0.1 and 1.0)

For the above said classes, the value of D_1 would be

$$\text{I } D_e = \sum_{i=1}^n d_{ei} = \sum_{i=1}^n (\alpha_{ei} - 1) W_i = \sum_{i=1}^n (\alpha_{ei} - 1)$$

$$\text{II } D_e = \sum_{i=1}^n (\alpha_{ei} - 1) \left[1 - \frac{i-1}{n-1} \right] = \frac{1}{n-1} \sum_{i=1}^n (\alpha_{ei} - 1) (n-i)$$

$$\text{III } D_e = \sum_{i=1}^{n_0} (\alpha_{ei} - 1) + \frac{1}{n-n_0} \sum_{i=n_0+1}^n (\alpha_{ei} - 1) (n-i)$$

$$\text{IV } D_e = \sum_{i=1}^n (\alpha_{ei} - 1) e^{-c(i-1)}$$

The results of the implementation of this model are given next. The attribute values are determined by experiments and system manuals. The minimum requirements have been decided by interviewing the system maintenance staff and the student users. However, a simpler, feasible approach to evaluation involves assigning of weights randomly to the attributes based on the individual judgement of the evaluator.

For ranking the candidate systems in this case, the attributes which are more important to the specific application in mind is first identified and then accordingly they are assigned relative weights by mere judgement and experience of the evaluator. These weights are so assigned that they are proportional to the relative importance of the attributes. For example in a particular case the weights are developed by having a maximum of 100 points. These are distributed in such a way among the features that they reflect the relative value of each feature. The value of such a process depends on the completeness of the features selected, their independence (they shouldn't measure the same thing), and the additivity of the values given to the different features. However, it is not

possible to be very accurate. If two or more systems are close in value, either of them could be the right choice.

After the weights are assigned, each system is to be studied and assigned a score (from 0 to 1.0) for each technical feature. This scoring will take care of both, the minimum requirement and the actual value of the feature i.e. if the minimum requirement of CPU time for one particular benchmark is 0.3 sec. then if for one of the candidate systems this time is 0.15 secs. then the score allotted to this feature of the system can be either 0.5 or something near about that depending on the evaluator's viewpoint as unlike a direct ratio in the previous methods.

Finally, each system is assigned a final figure of merit by summing the products of every technical feature's relative weight and the system's score for the technical feature. Stated algebraically, the figure of merit is

$$D_j = \sum_{i=1}^n w_i \times S_i$$

where W_i is the weight assigned to the i^{th} attribute and S_i is the systems' score on the i^{th} factor.

~~To calculate the efficiency of a system, the following can be used.~~

5. CASE STUDY

Case Study : Computer systems for educational use.

For the purpose of evaluation of a computer system for an educational environment, the major characteristics of interest can be classified as :

1. Architectural Qualities
2. Software Qualities

This identification would eventually lead to quantitative metrics. The case study will include only the Architectural qualities.

1. Architectural Qualities :

1.1 Accessibility : This characteristic indicates the extent to which the architecture permits control over its functional units. Two of the features which can be categorized here are :

1.1.1 Access to processor states : i.e. user state when only a subset of the instruction set can be executed and a supervisor state, for executing all privileged operations.

1.1.2 Security : This can be gauged by noting the provisions provided by the architecture to prevent unauthorized access to its components.

1.2 Addressability : The features of interest under this characteristic are :

1.2.1 Addressing modes : The different types and the flexibility of addressing modes provided by the architecture for efficient access to data structures.

1.2.2 Address range : This is the range of addresses a program can actively reference.

1.2.3 Address Spaces : The architectures' ability to support different address spaces like a

- . Program segment, where only "pure" code (i.e., read - only) can be stored. This feature is useful for supporting reentrant programs and multiprogramming.
- . Data segment for storing (temporary) data variables (i.e., random access).
- . Communications segment which is a common communication area that could be accessed by different processes to exchange messages.
- . I/O address space which can be assigned to I/O peripherals.

1.3 Compatability : The features supported by the architecture for upward compatability is measured by this characteristic. For example,

- 1.3.1. Undefined opcodes : By having an undefined opcode space, new instructions can be easily implemented in future.
- 1.3.2 Virtualizability : This measures the ability of the architecture to support a virtual machine concept, wherein a mechanism exists for a privileged standalone program to run as an unprivileged task and produce results identical to those it produces as a privileged program.
- 1.4 Conceptual Integrity : The quality of an architecture as exhibited by its, complete, coherent and consistent structure, so that special or exceptional cases are minimized, is indicated by this characteristic.
- 1.5 Context Switching : This characteristic measures the architectures' ability to support and handle changes in context (e.g. interrupts and traps, subroutines, process switch, etc.). Features that merit consideration here are :

1.5.1 Context Switching Overhead : The overhead involved in switching the processor state (e.g. context switch time, the amount of processor state information that needs to be saved, etc.). For real-time applications, Context switching overhead should be minimal.

1.5.2 Visibility to context switching : Factors of interest are :

- . The number of levels of exception allowed.
- . Nesting of levels of context switches, and
- . Provisions for masking under program/master control.

1.6 Efficiency : This is defined as the extent to which the architecture makes use of its resources.

Features that affect the architecture's efficiency are :

1.6.1 Compiler Complexity : The extent to which the code generation process of the compiler is simplified is a measure of the compiler's complexity.

1.6.2 Complexity of hardware : The hardware complexity of the architecture is determined by factors like :

- . Component minimization : This contributes to efficiency of space and speed, and the general robustness of the architecture.

- . Functional Modularity : This depends on the extent to which the architecture supports a modular design across its functions.

1.6.3 Cost of hardware : The efficiency of implementation of the architecture (i.e., whether it is cost-effective) is determined by the hardware costs.

1.6.4 OS Overhead : The overhead involved in supporting operating system functions (e.g., access control, address translation, etc.) has a significant impact on the overall efficiency of the system.

1.6.5 Performance : The performance of the architecture can be determined by measures like :

- . external job throughput (i.e. as visible outside the system)

- . raw speed of execution (i.e. the number of operations per unit time).

1.7 Feasibility : This indicates as to whether the architecture can be implemented using the state of the art technology (e.g. VLSI, microprogramming, hardwired, distributed functionality, etc.).

1.8 Human Engineering : This characteristic deals with the man-machine relationship and measures the extent to which the architecture contributes to a convenient user interface. Some of the factors that deal with this aspect are :

1.8.1 Accessibility : The ability of the architecture to permit a controlled access to its functional components.

1.8.2 Protection and Privacy : The extent to which the architecture contributes to the system protection from users and guarantees privacy to individual users.

- 1.8.3 **Responsiveness** : The extent to which the architecture is responsive to user demands.
- 1.8.4 **Runtime Support** : This deals with features like user friendliness, system messages, etc. that establish a good rapport between the user and the system.
- 1.9 **Modifiability** : This characteristic describes the extent to which the architecture is amenable for modifications in the systems hardware/software. Some features that relate to this characteristic are :
 - 1.9.1 **Device Independence** : This indicates whether the architecture treats its functional units (e.g. I/O, memory) as logical entities.
 - 1.9.2 **Documentation** : A well documented architecture facilitates modifications to the design and aids in debugging.
 - 1.9.3 **Microprogrammability** : The extent to which the architecture is microprogrammable, which could be used for augmenting the instruction set, modifying

the architecture etc.

1.9.4 Modularity : The extent to which the architecture support modular design.

1.9.5 Technology Independence : Implementation independence of the design so that advances in device technologies can be easily incorporated into the architecture.

1.10 Support for Languages :

1.10.1 Data format support : This describes the ability of the architecture to support different data formats efficiently with features like :

1.10.1.1 Functional units to handle different data formats, e.g. floating-point unit, decimal-unit etc.

1.10.1.2 Generic instructions with self identifying data.

1.10.2 Data Structures : The architecture's support of major data structures depends on factors like :

- 1.10.2.1 Access mechanisms : Efficient access to data structures is related to the flexibility of the addressing modes.
- 1.10.2.2 Correspondence : This is a measure of the degree of correspondence between the data structures (e.g. arrays) and their representation in the architecture. The greater the correspondence, the lesser the compiler complexity.
- 1.10.3 High-Level Programming : This indicates whether the architecture can support very high-level programming style, like functional languages (e.g. pure LISP), logic programming (e.g. PROLOG), etc.
- 1.11 Testability : This defines the extent to which the architecture facilitates testing procedures.
 - 1.11.1 Hardware Testability : It depends on access to internal components, which is restricted by architecture design and modularity of design, which enables testing of modules independently.

1.11.2 Software Testability : Software debugging is aided by testing for anomalous conditions like overflows, invalid data representations, I/O errors, bound checks etc.

6. CONCLUSION

Computer performance evaluation has been a subjective study. However an attempt has been made to reduce the subjectivity involved by a quantitative evaluation strategy.

The methodology consists of identifying a set of attributes in terms of which candidate computer systems will be evaluated. The weighted difference between the requirements and the extent to which a system possesses an attribute determines the figure of merit of the system which gives a quantitative "measure of goodness" of the system.

The evaluation of the architectural qualities of a computer system for educational use is given as a case study.

A subjective assessment of the suitability of the HP1000 architecture for educational use is also given. The figure of merit assigned to the HP1000 was 80%.

APPENDIX I

EVALUATION OF THE H.P.1000

ARCHITECTURE FOR EDUCATIONAL USE

This Appendix gives a subjective assessment of the suitability of the HP1000 for use in an educational environment. The system is evaluated against each feature mentioned in Chapter IV.

1.1 ^{cc. 1} ~~Access~~ibility :

Access to processor states : The HP 1000 prohibits execution of privileged instructions (mapping instructions and all I/O instructions except those referencing select code 0¹, the CPU status register and the overflow register). This limits control of I/O and mapping operations to the operating system or other privileged programs.

Security : Provides for the detection of unauthorized access to memory by a memory protect logic on the CPU. Protects memory on a page-by-page basis against alteration or entry by programmed instructions, except those involving the A and B registers. A memory protect violation instruction will interrupt the CPU and the address of the violating instruction will be saved in a register on the memory controller card,

from which it can be made accessible in the A or B register by a single Assembly language instruction.

An unimplemented instruction interrupt is also generated when the CPU signals that the last instruction fetched was not recognized.

1.2 Addressability :

Address modes : The instruction set provides for Direct, multilevel-indirect, single word, double word and register implicit, indexed and indirect indexed addressing modes.

Address Range : The maximum memory expansion is upto 10.5 K pages of 2048 bytes/page. The memory structure allows 32 pages of 2048 bytes with direct access to current or base page (page 00), indirect or indexed access to all other pages.

Program Segment and Data Segment : The architecture allows memory protection for write or read or read/write on a page by page basis.

1.3 **Compatability :**

A number of undefined op.codes exists which can be used for the implementation of user defined instructions as it supports user microprogrammability by an optional writeable control store card and an easy to use Pascal like paraphraser.

1.4 **Conceptual Integrity :**

The HP1000 has a consistent structure so that exceptional cases are minimized.

1.5 **Context Switching :**

Context switching time : 3.7 to 1.3 μ sec.,
4 μ sec. typical.

This is when there is no DMA interference. It offers I/O device interrupt priority which depends upon the I/O interface card position along the backplane with respect to the CPU card.

Interrupt Masking : The I/O logic includes an interrupt mask register which provides for selective inhibition of

interrupt from specific interfaces under program control. This capability can be programmed to temporarily cut off undesirable interrupts from any combination of interfaces.

1.6 **Efficiency :**

The efficiency of the HP 1000 can be rated quite high because the compilation speed, external job throughput and the raw speed of execution are too good for an educational environment.

1.7 **Feasibility :**

HP 1000 is microprogrammed.

1.8 **Human Engineering :**

The HP1000 is very user friendly. Its functional components are easily accessible and provides a convenient user interface. Its response is adequate for an educational environment.

1.9 Modifiability :

Device Independence : Its functional units (I/O and memory) are not treated as logical entities but there are separate instructions for I/O and memory references.

Dynamic tuning : It is supported by an optional control store board. This gives the user the ability to convert software routines that are frequently used or especially time consuming to microcoded routines that typically run 2 to 10 times faster.

1.10 Support for languages :

It has bit manipulation instructions,
byte manipulation instructions,
word manipulation instructions,
dynamic mapping instructions,
double-integer,
single-precision floating point,
double-precision floating point
single precision scientific instruction set,
vector instruction set both single and double
precisions.

Subroutine linkage : There is no specific method for parameter passing.

1.11 Testability :

They have microcoded and macrocoded self tests which check the CPU, memory, and the I/O masters of installed interfaces, either automatically on power-up or when requested by an operator via the virtual control panel.

1.12 Support for Systems Programming :

It has some supports like TBS, CBS, SBS.

: A7 :

By using the class I graph for assigning the weights wherein all attributes have the same weights = 1

We have the following

$$D_1 = \sum_{i=1}^{12} (\alpha_{1i} - 1)$$
$$= (\alpha_1 - 1) + (\alpha_2 - 1) + \dots + (\alpha_{12} - 1)$$

$$\alpha_1 = 1.8$$

$$\alpha_7 = 1.7$$

$$\alpha_2 = 1.7$$

$$\alpha_8 = 1.9$$

$$\alpha_3 = 1.9$$

$$\alpha_9 = 1.7$$

$$\alpha_4 = 1.9$$

$$\alpha_{10} = 1.8$$

$$\alpha_5 = 1.8$$

$$\alpha_{11} = 1.8$$

$$\alpha_6 = 1.8$$

$$\alpha_{12} = 1.8$$

$$6(.8) + (.7)3 + (.9)3$$

$$= 4.8 + 2.1 + 2.7 = 9.6$$
$$\frac{9.6}{12} = 0.8$$

% performance = 80%

BIBLIOGRAPHY

1. Computer Systems performance Evaluation
By - Domenico Ferrari, 1978 Prentice-Hall, INC.
2. Computer Performance Evaluation
By - Philip J Kiviat, 1976, Online Conferences
Limited, Uxbridge, England.
3. Software Design Strategies
By - Glenn D. Bergland, Ronald D. Gordon, Bell
Laboratories, Murray Hill, New Jersey.
4. A Computer Perspective
By the office of Charles and Ray Eames.
5. Fundamentals of Computer Science
By Andrew J.T. Colin, 1980, The Macmillan Press Ltd.
6. Timesharing System Design Concepts
By Richard W. Watson, 1970, McGraw-Hill Book Co. Ltd.
7. Timesharing Systems
By G.M. Bull and S.F.G. Packham
1971 McGraw-Hill Book Company Ltd.
8. Fundamentals of Operating Systems
By Donovan

9. Compiler Construction
By F.L.Bauer and J.Eickel, Springer-Verlag
10. Computer Organization
By V.Carl Hamacher, Zvonko G. Vranesic 1978,
Mc-Graw Hill Book Company Ltd.
11. Principles of Compiler Design
By Alfred V. AHO, Jeffrey D. Ullman. 1979,
Addison-Wesley Publishing Company.
12. Workshop on Microprocessor Architecture and Systems,
Computer, pp. 49-51, Sept. 1976.
13. Fault Tolerant System Workshop, Research Triangle
Institute, Dec. 3-5, 1975.
14. S.Chang, "A model for distributed Computer System
Design", IEEE Trans. Syst., Man., Cybern.,
pp.344-359, May 1976.
15. 'A Fremework for the Quantitative Evaluation of
Distributed Computer Systems'. - Mario J. Gonzalez.
JR., and Bernard W. Jordan, JR., pp.1087-1094,
Dec. 1980, IEEE Trans. Computer, TC.