Chapter 5

Overall File-System Evaluation

The purpose of computing is insight, not numbers

Richard Hamming

The number of computations without purpose is out of sight

Author's corollary

We have, up to this point, analyzed the technical performance of file systems. The parameters developed in the previous chapters have yet to be translated into financial costs and economic benefits before we can provide management with data which will allow them to judge the appropriateness of a database which uses computer-based file systems.

We have to assume now that the decision to install a database system is made on rational grounds. Other considerations that have led to the use of computer systems have been based on the expectation that rapid access to information would eliminate existing operational problems. This is unlikely to happen. Decisions have also been made in imitation of leaders in the same segment of industry. Often substantial pressures are exerted by the sellers of computing equipment.

In the scope of this book we cannot evaluate all intangible elements which are part and parcel of decisions to install computers, but will rather present a model of a functional evaluation. A number of decisions of a nontechnical nature remain, specifically in the area of attributing financial or social benefits to system performance.

The Process of Evaluation Most benefit factors can only be estimated, but the fact that they are *estimates* does not at all diminish the importance of assigning quantitative values to all important parameters in a system design. The use of specific, documented, and consistent values for these quantities allows analysis of the decision and the comparison of system and nonsystem alternatives. When design decisions are made which are based on unrecorded assumptions, the probability of design errors is high, and correction of assumptions to improve the design is impossible.

If we have to make very many assumptions, then our conclusions will be mainly of a relative rather than absolute nature. This state will become obvious during the estimation process. Figure 5-1 provides an overview of the analysis process.

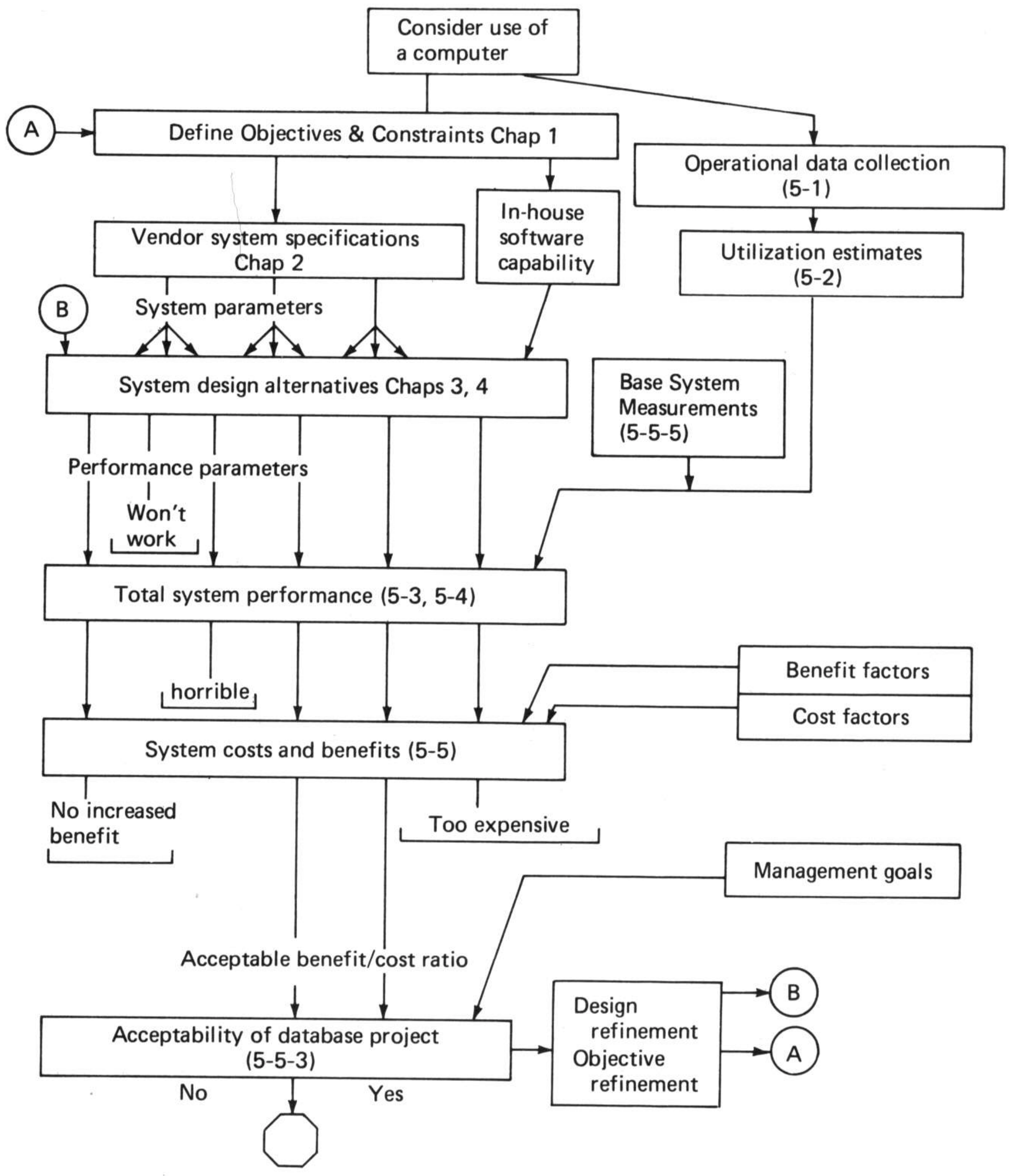

Consider use of a computer
A
Define Objectives & Constraints Chap 1
Operational data collection (5-1)
In-house software capability
Vendor system specifications Chap 2
Utilization estimates (5-2)
B
System parameters
System design alternatives Chaps 3, 4
Base System Measurements (5-5-5)
Performance parameters
Won't work
Total system performance (5-3, 5-4)
Benefit factors
Cost factors
horrible
System costs and benefits (5-5)
No increased benefit
Too expensive
Management goals
Acceptable benefit/cost ratio
Acceptability of database project (5-5-3)
No
Yes
Design refinement
Objective refinement
B
A

Figure 5-1 The system-analysis process.

Section 5-1 of this chapter deals with the estimation of system use, whereas the three sections following will deal respectively with system benefits, storage costs, and production costs. Section 5-4 includes considerations of multiprogramming, use of multiple processors, and data distribution. Section 5-5 presents the problems of integrating these factors and ensuring the robustness of the final results.

We will frequently relate the analysis to the seven performance parameters obtained in Chap. 3 and listed on the inside covers. These parameters describe the following:

1 The required data storage: $n_{total}R$

2 The expected search times: T_F, T_N, T_X

3 The expected update times: T_I, T_U

4 The time required for reorganizations: T_Y

5-1 THE ESTIMATION OF SYSTEM USAGE

In order to extract benefits out of a database, it has to be used. The benefits which may be obtained, the effectiveness with which they are obtained, and the cost of the database operation are all related to the load placed on the system. The load consists of demands on data storage and on operations requested by the transactions.

Data Volume A primary aspect of the load is the volume of stored data required for an effective operation. In many environments the volume of data is expected to increase steadily. The estimation of the load placed on the file system in terms of volume is controlled on the number of records n, the total number of attributes a, and the average number of attributes in a record a'. The lengths of the value fields V and the attribute description fields A also play a role.

The various file designs will combine these common parameters in different ways, so that at this point we are not prepared to evaluate the file size, since the design decision is yet to be made. If we cannot contain our anxiousness, we may quickly check the magnitude of the storage demand presented by a pile organization of the proposed data.

Data-Retrieval Load The retrieval of data elements is the central objective of a file system. A transaction may require data from more than one file. We stated earlier that a collection of related data forms a database. If it is made accessible to a set of transactions, we can speak of a database system. The primary objective of database systems is the synthesis of data and, hence, the creation of information. Some measure of what this means in terms of the utilization of the file system is required. We consider first the load on systems by data requests for services for operational purposes.

The operational load is provided by an estimate of the number of service requests to a file made by the transactions in a given period. The distribution of these requests over a daily or monthly cycle should also be estimated to determine periods of high demand. The load and time of busy intervals are used to verify that the proposed system is adequate at those times.

Example 5-1 Load for an airline reservation system

We wish to design a booking system for a small airline. We have studied the operation and can summarize the result as follows:

> The number of salable items we have is determined by the number of flights between two adjacent points and the number of available seats. With 3 aircraft of 70 seats each, which complete 20 flight segments per day, nearly 30 000 units are for sale each week. The average ticket sale represents 2.7 units because people fly more than one segment and use return flights. The average number of people flying and booking together is 1.61. A survey shows that 20% of all reservations are subsequently changed. During the process of selling a ticket, typically three possible flight schedules are explored.

The load L on the system is stated in terms of fetch and get-next accesses for ticket sales

$$12L_F = \frac{30\,000}{2.7 \cdot 1.61}\,1.20 \cdot 3 = 25\,000$$

$$L_N = 30\,000 \cdot 1.20 \cdot 3 - L_F = 83\,000$$

We have to note for the subsequent file-design phases that additional segments of a flight schedule and additional people on a flight are accommodated through get-next searches.

The estimates used in the example may be rough, but it is important that they have been made and have been written down. A change in projected system usage can be accommodated only if the original assumptions are known. In addition, it is desirable to redo the complete evaluation for both high and low usage estimates in order to be assured that the effects of misestimation are neither fatal in terms of cost nor disastrous in terms of performance.

Load for Information Retrieval For a database system which is to produce primarily *information*, rather than produce data to control sales of services or equipment, the estimation of the load is considerably more difficult. The extent of its use is mainly dependent on user satisfaction. Satisfaction in turn depends on human factors, utility, and performance.

A retrospective health-status report on a patient loses its value quickly if the effort to obtain it is greater than the desire of the physician to have the past record available during the encounter with the patient. Such a health report only occasionally contains information which is not obvious to the physician at the time of the visit. The average benefit may be quite low.

For information systems the estimation of load may require investigations of similar systems. If no similar situations exist, the operation of a *pilot model* can provide data for load projections. In a pilot model, different technology can be employed, since it is allowable to operate at much higher unit cost per query in order to minimize the initial investment, while obtaining performance parameters which match the intended final form of the proposed system.

A prediction based on a pilot model requires reflection and insight when the pilot model was operated with a limited database. The designer can only conjecture how system benefits, and hence usage patterns, are related to file size. A careful analysis of the information-producing potential and the power this information provides for the users may be required. Usage patterns also change over time. A flexible design is needed to allow the system to grow and adapt.

The outcome of the above considerations should be the frequency of all types of retrieval:

Fetch requests L_F
Requests for *next* records L_N
Exhaustive searches L_X

both in terms of averages over long periods (days, weeks, or months), as well as the maximum rates for short, high-intensity periods that may be expected.

Update Load The frequency of updates of the system also will be considered in the estimation of system volume. We wish at this point to break down the update frequencies into types of tasks that represent:

Addition of a new record L_I
Change of a field in a record L_U
Extension of a record L_Z
Deletion of a record L_D

In the airline example introduced above, each unit of the flight inventory has to be updated for every successful sale and two updates are needed for a change.

$$L_U = 30\,000(1.00 + 2 \cdot 0.20) = 42\,000$$

In order to keep track of the passengers, a separate file `P` may be maintained. A new record is inserted for every new passenger (36%), and this record is extended with every additional purchase or change made.

$$L_I(\mathtt{P}) = 0.36 \cdot 30\,000/2.7 = 4\,000$$
$$L_Z(\mathtt{P}) = 0.64 \cdot 30\,000 \cdot 1.2/2.7 = 8\,540$$

All these values are weekly rates; a busy time to check separately may be Friday afternoon.

Eventually the load parameters have to be estimated for all files, all operations, and several activity periods.

Depending on the type of file organization, we will associate each of these load frequencies with the appropriate performance parameters.

If we consider only our six basic file methods, we find the relationships shown in Table 5-1. We note that the load parameters $\{L_I, L_U, L_Z, \ldots\}$ do not map directly into the file performance parameters $\{T_I, T_U, \ldots\}$, since not all actions are directly supported by all file organizations. The assignment may also depend on the expected response time. We have learned enough about the various file organizations to know that some file types are not suitable for certain on-line operations.

Table 5-1 Parameters Appropriate for Update Evaluation

	Pile	Seq.	Index Seq.	Index	Direct	Ring
L_I	T_I	T_I	T_I	T_I	T_I	T_I
L_U	T_U	T_I	T_U	T_U	T_U	T_U
L_Z	T_U			T_U		T_U
L_D	$T_U - T_I$	T_I	T_U	T_U	T_U	T_U

File Maintenance The frequency of reorganization is largely a function of system design and file dynamics. Reorganization may be combined with periodic exhaustive processing so that

$$L_Y = L_X$$

In other cases, reorganization can be specified as a function of the past update activity. Section 6-4-3 presents an analytic procedure to determine reorganization frequency.

The small airline of our example may clean up its passenger list once a week as part of a market summary but reorganize its inventory file as needed to keep access rapid. After 20% update activity, access to the weekly inventory file is measurably slowed down.

$$L_Y(\texttt{P}) = L_X = 1$$
$$L_Y(\texttt{I}) = (L_I + L_U + L_Z + L_D)/(0.20 \cdot 30\,000)$$

We will apply these load parameters to a file system in Sec. 5-4 but will first discuss the benefits to be obtained from the transactions which give rise to these load factors.

5-2 ANALYSIS OF SYSTEM BENEFITS

The first performance parameter (nR), which specifies the quantity of data to be stored, is not directly associated with any system benefits. This remark is meant to reinforce the realization that, except for some historians, the mere collection of data is not an end in itself. The benefits of having data arise from the use of data. The use of data may be vague and distant, but its probability needs to be evaluated and measured. The availability of stored data is, of course, a prerequisite for use of a database system.

5-2-1 Benefit Quantification

The value of a completed transaction is best quantified in financial terms. In the case that the use of the system is associated with sales, this value can simply be a fraction of the profit expected from sales.

In other cases, the benefit may be established by questioning the potential users to determine what they would be willing to pay for the service. If the situation is one where money does not provide a basis of measurement, a comparison of the desirability between this service and other services whose cost is known can be used to provide measures of service value.

An example is from the physician's office, where the availability of a past medical record may be compared in terms of diagnostic power with another laboratory test.

We can only get into trouble if we accept in this context a statement that the service is essential, so that its value seems infinite.

Frequently benefits are expected from lower personnel costs. Here we have to assure ourselves that these benefits are realizable, and even if they are, that they are socially desirable. Realization of personnel economies is difficult to achieve if fractions of the effort of a number of individuals are being saved. If new jobs have to be created out of portions of partially eliminated positions, we have to ensure that the new position is viable. Replacement of untrained personnel by fewer but more expensive trained people is also questionable. Problems can occur if the computer handles routine services but not some task which is to be done during a short time interval.

If a computer in a banking office allows two people to handle the customers previously served by four people but does not assist in the reconciliation of funds done daily between 3 and 5 P.M., then changes in the daily procedures will be needed.

The cost of readjustment of operations also has to be included before summing up the benefits.

Expectations of future benefits are often based on cost containment. This implies that while there is no net saving now, the new system will allow growth at a smaller increase in cost than can be foreseen otherwise. It is obvious that such reasoning requires an even more careful benefit analysis, since the rate of expected growth is always uncertain and affects, when achieved, many aspects of an organization beyond the area of computing systems.

Frequently the benefit of database operations diminishes gradually with increased response time. Figure 5-2 shows the estimated loss of benefit at a city airline counter where the customer has a choice of airlines and much of the business is on a walk-up basis. The logarithmic scale on the abscissa is used in these sketches because we are dealing with such a wide range of time values.

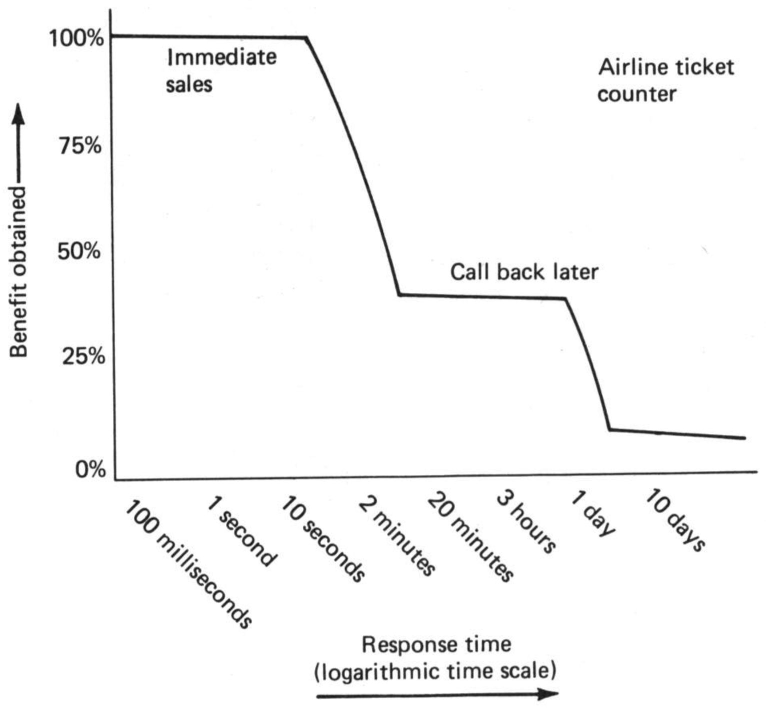


Figure 5-2 Benefit realization.

Real-Time Operation If the response time of the system is poor, the data may arrive too late to have the desired effect. This presents us with the *real-time* constraint. The value of this constraint depends on the application and can vary from milliseconds to months.

Table 5-2 lists some estimates of real-time constraints in various areas where computer systems with databases have been employed. The validity of some of the limits may be debatable, but for each of these categories, there are instances of systems which provided a longer response time and had problems in operational acceptance. The total response time includes the entry of the request, the computer system activity, and the presentation of the response. Only a fraction of the times shown is available for file operations. We also have listed in Table 5-2 the typical quantity of data expected corresponding to one request in these areas.

The need to keep within real-time constraints may mean that we will not be able to consider in some decisions all the possible information, since the time to fetch the data would be excessive. Decisions taken under operational real-time constraints hence may be less than optimal, but yet better than decisions taken without any data. It is possible that there are data elements which, although they are conceptually useful, do not actually contribute to the decision-making process of the user.

Table 5-2 Real-Time Constraints and Data Volume

Area	Value of constraint	Reason for size of constraint	Search volume/ request(bytes)
Annual accounting	Month	Annual report cycle	10^4
Full auditing report	Week	Audits are preannounced	10^3
Inventory report	2 days	Ordering cycle is a week	10^3
Payroll	2 or 3 days	Accepted by employees	10^3
Stock trade	1 day	SEC requirement	10^3 to 10^4
Bank account update	Overnight	Expectation by customers, early detection of problems	10^5
Audit inquiry	Few hours	Request has to be handled via employees of firm	10^2
Casting agency	10 min	Multiple contacts have to be made in an hour	10^2
Airline-ticket sales	1 to 5 min	Patience of passengers	10
Real-estate sales index	$\frac{1}{2}$ to 2 min	Salesperson does not want to lose continuity	1 to 20
Grocery counter credit check	3 to 20 s	Goodwill	1
Item from medical file	2 to 10 s	Operational cycle of MD	50 to 10
Positioning of a mass spectrometer	100 μs	Vaporization of sample	2

Effectiveness versus Benefits Systems are not always placed into operation with clear or measurable expectations of benefit. When, for instance, a certain service is mandated by law, only the effectiveness of the implementation can be determined; the benefits of the law have presumably been established already.

If no reasonable measure of benefit can be found, the base benefit is set to zero. It is still possible to proceed through the arguments which follow. Many considerations will reduce the expected benefits. Those reductions will have to be estimated in absolute terms, and by the end of this section, only negative benefits will be left. This still will allow the integrated system comparisons in Sec. 5-5 to be made, but the conclusions will have to be presented to management in a relative sense. This approach, termed cost-effectiveness analysis rather than cost-benefit analysis, is common in government operations.

Cost-benefit analysis attempts to obtain quantitative data so that *go/no-go* decisions can be made.

Cost-effectiveness analysis permits evaluation to select one of several system alternatives which will achieve a desirable goal.

Any system has to achieve a certain level of quality of performance in order to become effective. The benefits of the transactions can be fully realized only if the database system performs ideally. The topics following will discuss these effectiveness factors in terms of the performance parameters to be obtained.

5-2-2 System Utilization

The operator effort required to use a system diminishes its benefits. If there is a full-time operator the basic costs are easy to determine. A slow or awkward system will increase the basic cost per transaction. If the system is operated by people who have additional and perhaps more important tasks, the operability of the system will be an essential component of its usage cost.

Measures other than response time are not easily quantified. Counting of keystrokes, as done when comparing computer editors, seems futile. The effort required from the user is easy to underestimate, and predictions by system designers have to be substantiated during pilot operation with the intended users.

Direct time measures of the effort to use the system include

1 The time required to formulate queries
2 The time spent in order to describe the format of desired outputs
3 The time needed to enter the search parameters into the system
4 The time required for the actual processing of the request
5 The delay incurred for display or printing of the result

Comparison of these times with existing manual systems can provide some idea of the relative demands made on the users. Our performance parameters help mainly to establish the delay in item **4**, and some of that time can be overlapped with entry **3** and presentation **5**.

Consideration of the effort required to use the system is particularly significant where there is direct use of computers by professional users such as managers, lawyers, physicians, scientists, and researchers. Here one has to avoid a diminution of the time a professional has available for work of primary interest.

In a number of instances database systems intended for direct use by professionals have had to be augmented after installation with clerical personnel at the terminals. The costs and delays related to the use of intermediaries to operate the terminals seriously affect system economics.

Another component of the personnel effort is the amount of training required to use the system. The cost of training will be relatively low for frequent users, but if we expect occasional users to benefit from the system, the amount of requisite training should be minimized. The additional effort of designing database systems so that the user needs little indoctrination can be well justified in terms of overall system cost.

Personnel Cost The effect of a system on personnel costs is an important part of proper system design. It is not unusual that the cost of terminal operators and time spent by people preparing and receiving data exceeds the cost of the hardware used by a system.

Various approaches can be followed to determine the relationships between performance and response time. Industrial engineers have measured user response time at the terminal. It has been found that different work habits will evolve depending on the interface that the system provides.

As an example, we will use the financial collection office of a hospital where unpaid accounts are reviewed for follow-up. The computer system is used to provide an up-to-date status report so that all payments, credits, adjustments, and so forth, will be known before a collection effort is made.

A system which responds instantaneously relative to the reaction time of the clerk will allow a continuous performance of the collection task as shown in Fig. 5-3.

If the system does not respond quickly, some waste of the operator's time will occur. The operator delay is generally greater than the system delay which caused it, because the operator's flow of consciousness is disturbed. If the delays become significant, they provide reasons for trips to the coffee machine, checkbook balancing, and such.

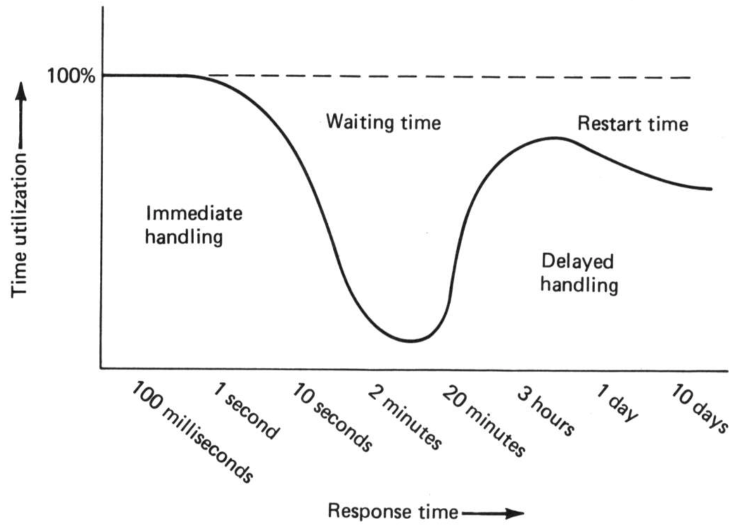


Figure 5-3 Personnel-time utilization.

At some point, the operators will adopt a different mode of operation; they will make a number of requests simultaneously, and during the time the system needs to produce the records, the clerk will process the previous batch. Now the wasted time is much less, since it consists only of an initial starting time and an increased frequency of attention switching. If the delays exceed 8 hours, yet other work procedures will be used. If the delays exceed a week, it is probably not worthwhile to use the system at all.

5-2-3 Data-Content Maintenance

Updating of the database is often the costliest component of the aggregate system. In some cases the data, or a portion of it, can be gathered automatically. Point-of-sale recording devices can provide the input to control an inventory when the sale occurs, and similar devices are becoming available for other areas. A large portion of the information in databases still is entered manually, either directly via computer terminals or indirectly via cards, tapes, or optically read documents.

In the evaluation of data-entry costs, we cannot consider only how much it costs to enter a data element. The proper question is how much it costs to enter a data element correctly. The cost of detection and correction of an element entered incorrectly, as well as the cost of the havoc the erroneous value might have wrought while it was part of the database, has to be added to the data-entry cost.

Errors on data entry can be classified into errors occurring before, during, or after the entry of the data into the computer.

Before Entry The reduction of errors made prior to transcription into a computer is a problem largely outside of the scope of this book. If the transcription is close to the source of data, the frequency of such errors will be less. A proper overall system design can contribute greatly to minimization of errors. Coding methods appropriate to a given task are discussed in Chap. 14.

During Entry The reduction of data entry errors is strongly related to the quality of the transcription device and its operation. The quality, i.e., the delay time and its consistency, of the system response to an update has an effect similar to the effect discussed earlier when searching for data. Here poor system performance actually can cause negative productivity, as shown in Fig. 5-4.

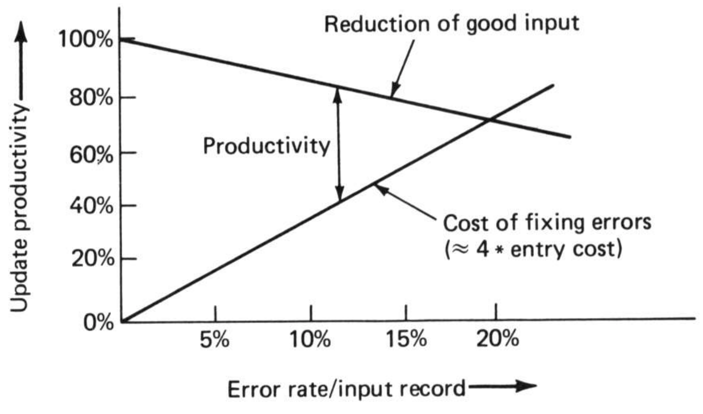


Figure 5-4 Error rate versus productivity during data entry.

This figure still excludes indirect costs due to erroneous data which entered the database. If, because of wrong information caused by a data-entry error, undesirable actions are performed, the cost of the error may cancel all benefits obtained from the system over a long period.

Through an erroneous inquiry, we could, for example, have committed our company to the delivery of stock which was already sold. Replacement stock to fulfill the obligation has to be obtained at a premium.

Feedback The most important principle to reduce errors on data entry is the use of *feedback*. Feedback loops can be provided to the operators during data entry. to the data collectors by regular reports and summaries, and to higher level personnel by reducing data to trends and graphs. It is best if the feedback does not appear in the exact form that the data were entered, and that the output includes some additional information.

Entry errors due to mistyping of terms can, for instance, be reduced drastically through the use of input devices providing selection capability. Display screens on a CRT terminal that are formatted like a menu or a form with a number of listed choices require only an indication or a check mark for the entry of a data element. Feedback of the selected value is provided on the screen by listing the name of the data element in full, so that mis-hits will be recognizable. Selection is sketched in Fig. 10-5.

When codes or identification numbers are used to enter data, the feedback on the display, in the form of plain language, of the meaning of the entered code can aid significantly in error control. Feedback of numeric values which have been entered in a well-formatted and legible display is also helpful. Where the customer is present, the feedback can be shown to the customer for verification. In many cases a second display terminal is employed for this purpose.

Use of feedback reduces the error rate per data element; what the effect is per unit time is not well known. The problem of correct entry of arbitrary alphabetic text or large numbers is not solved by this approach. Numeric data entry correctness can be verified by using control totals. A batch total, prepared when the source data are collected, is compared with the sum of the entries, and any discrepancy is reported.

After Entry Database systems may include automated input format verification procedures, as well as comparisons with previous data recorded or recently entered. Ranges of value linits of attributes may be kept with the database. In many instances, even for attributes whose data values can have a great range, incremental changes, say, month-to-month values for a particular item, will not change more than a few percent.

If an error can be detected while the source of the data still is available, the cost of correction may be only a few times the cost of the original data entry. If errors are found much later, considerable time may be wasted in the retrieval of a source document or by having to refer back to the supplier of the data.

When response messages are designed, the various message texts should have significant differences in the initial words of the message, so that less frequent, and hence important error messages are immediately distinguished from repetitive confirmations. An audible error signal (see Fig. 14-1) can be very helpful when visual-display terminals are used to enter large quantities of data.

An additional measure to catch errors is to report to the originator of the data the data values entered. If, for instance, an order has been received by telephone or mail, the system can print a confirmation, in plain language, for inspection by the customer. The purchaser then may realize that you are planning to deliver next week `501 adders` instead of `50 ladders`.

In many instances data are not entered directly into computer systems. A common indirect entry method is the sequence of keypunching and verification (by rekeying and comparing), followed later by entry of a batch of data into the system. Indirect data entry is often less costly and reduces dependence of system performance during data entry. On the other hand, the detection of errors and feedback to the operators is deferred so that the cost of correction can be very high and repair of data frequently is omitted, knowingly or accidentally.

The decoupling of entry personnel from the computer system does avoid waste and frustration due to poor reliability and inconsistency of performance found in many systems. Checking of data for consistency may be done only during periodic report generation. Systems which do not provide feedback and incentives for error correction are doomed to eventual disuse. Bad experiences by users, who attempt to obtain information from a database and receive obviously incorrect data, can easily cause rejection of positive aspects of automation as well.

Auditability In some situations the problem of improper action caused by poor data may be so serious that one should not allow an error correction to be made by an update of the record containing the error. Instead a correction will be entered as a new record whose value will replace the previous value for a normal inquiry.

If a problem develops, we want to be able to go backward in time with a query of the form

"`What was the value for` x `that the system would have provided at a time` t `and date` d`?`"

and retrieve the erroneous value then in use.

An example of this requirement can be found in the maintenance of medical records where errors in actions resulting from poor data can cause great harm and invite legal claims of malpractice.

A general solution to this problem is to routinely identify *all* data elements with a *timestamp*. While this would in principle double the required storage space, the use of abbreviation techniques can reduce this factor. The availability of a time identifier with every data element can also help in the processing and management of data. Older data can be conveniently archived now, so that demands on high-performance storage devices may actually diminish.

Once correct data are entered into the system, we assume from the point of view of this chapter that they are safe. The chapters on data security (11 to 13) will deal with this problem on a technical level.

5-3 DATABASE STORAGE REQUIREMENTS

The operation of a computer requires personnel and hardware. We will concentrate in this chapter on the elements which relate to the load and performance factors that were developed in Sec. 5-1. Broader issues are discussed in Chap. 15.

In order to simplify the procedure, we will use methods which can provide rough initial estimates of load on a given system. These values can be refined later when the system being designed has taken a more definite form. This section will concentrate on the storage-related equipment, whereas Sec. 5-4 will discuss the processing-related system components.

5-3-1 Storage Utilization

When files undergo frequent and dynamic changes, not all the available file capacity can be utilized. Depending on the file system used, considerably more space may be needed. The density factor u_D is used to adjust the expected storage cost throughout the design process. The author has seen values for u_D ranging from 90% (stable files in an environment which provided good space-allocation capability) to 25% (dynamic files in a structurally limited environment).

A low value of u_D does not necessarily imply an undesirable file design. In files using direct access, for instance, available space will considerably enhance operational performance. In addition, there will be wasted space W because of unspanned packing of records into blocks, which reduces the number of blocks needed for some records.

The total storage capacity required for the files making up a database is

$$D = \sum_{allfiles} n\frac{R+W}{u_D} \qquad 5\text{-}1$$

Some additional storage capacity is needed for operating systems and programs. The parameters n, R, W, u_D are probably different for each file, but in this chapter we consider storage as if it were a single file to keep the notation simpler. The value D provides the initial estimate of the storage needs for a system.

5-3-2 Components

The primary hardware components of a system to provide storage of data are of course the storage devices, typically disks, themselves. We will in this section also consider the devices required to attach disks to the processors, *controllers* and *channels*. The performance and cost of a complete storage system are affected by the performance and costs of controllers and channels as well as the performance and costs of the disks.

Databases often require multiple disks and somtimes multiple controllers and channels as well. If the quantity D exceeds the capacity of one of the available disk units, multiple units will be employed. The value of the storage parameter $D_{available}$ is rounded to an integer multiple of the capacity of the devices to be used, unless we can convince others to purchase the remnant.

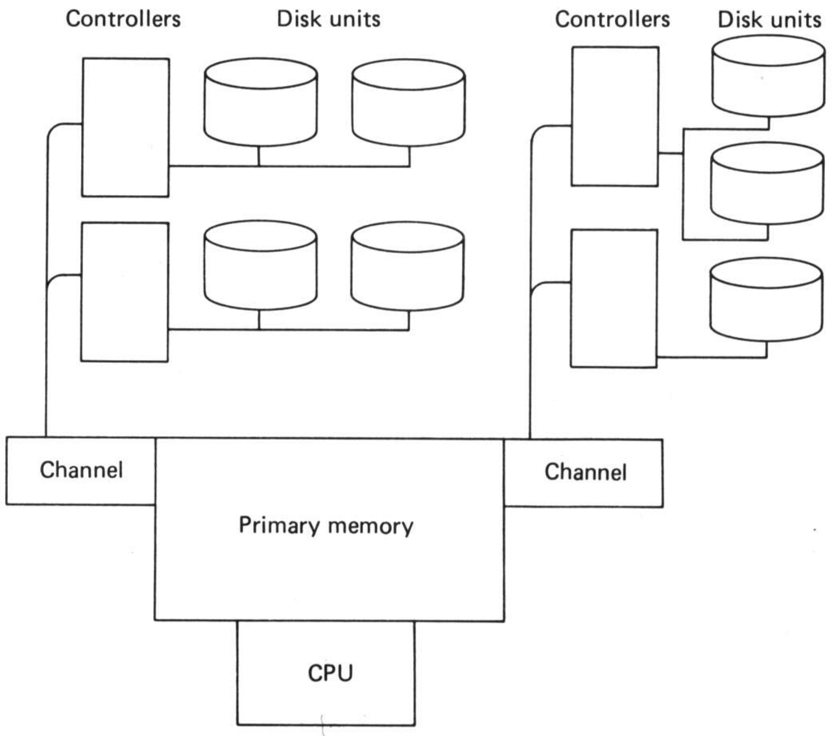


Figure 5-5 Storage-system components.

If the entire computer system is to be selected for a specific database, there may be much freedom in equipment selection. In general it is desirable to select storage devices so that one or few disks will satisfy the storage needs without the complexity of many units. Backup and archiving capacity is also needed now.

In most computers, disks are not directly coupled to the processors but require a *controller*. A controller may be able to handle 2, 4, 8, or 16 disk units. If many disks are to be used, there will be more than one controller. On the larger computers, controllers are connected to the processor by channels, which provide the data path into the processor memory.

Multiple *channels* may also be used, not so much in order to provide the connection capability, but rather to provide multiple simultaneous data access paths, giving a greater bandwidth into the computer. Figure 5-5 sketches the organization of the hardware components for a storage system.

5-3-3 Distribution and Replication of Data

Multiple processors are found in an increasing number of computer systems. When the processors are *closely coupled*, the storage can be shared, so that the data can be distributed. The benefits of having multiple processors, from the database point of view, are increased computational capability and backup in case of processor failures. If protection is required for storage-device failures, storage needs to be replicated. Replication is common when processors are not closely coupled.

When the processors are *remote* from each other, the storage facilities are not shared in the same manner. Access to remote sites is indirect, via communication lines and the remote processor. We do not find direct connections to remote storage devices useful, unless the storage device is used for remote input or output, and this situation is not appropriate for databases.

Distributed operations distinguish local and remote access, and transactions may be decomposed into subtransactions containing requests for distinct sites. A local request is processed as before, but a request to access remote data has to be communicated first to a remote processor. That processor obtains the result and returns it to the originator.

The capability of remote lines is an order of magnitude less than that of disk devices, i.e., btt_{comm}¿¿btt_{disks}, so that the balance of a system design changes considerably when it is distributed. In order to overcome slow access to remote sites, some of the data may be *replicated.* Replication increases the cost of update but improves retrieval dramatically. Data that are frequenty read and infrequently updated are prime candidates for replication at the sites where the queries originate. If replication is used, the storage demands D increase.

With these considerations, we can estimate the types and number of components needed to satisfy the storage requirements for the database. Since the actual design parameters are not yet known, one can obtain a range of devices needed by establishing a range of the storage capacity needed, estimating the requirements at a low level of waste, free space, and redundancy (say $W = 0, u_D = 80\%)\rightarrow D_{low}$, and at a high level (maybe $W = R, u_D = 25\%)\rightarrow D_{high}$ of these conditions.

5-3-4 System Choice

Once the range of storage requirements is defined, we can limit the evaluation of alternative systems to an appropriate range. Since a similar range of disk devices is available from many manufacturers, we can concentrate initially on types of storage units, and avoid selection of a specific vendor. We will select from the alternatives a file configuration which promises minimum cost. Figure 5-6 shows cost estimates for three storage-system hardware combinations related to capacity.

Discontinuities exist in these cost functions because of disk type alternatives, controller increments, and choice of reasonable channel assignments. Since an approximate range for the storage requirements has been established, only a segment of the curve is of interest. Three outcomes are possible:

1 The actual storage cost does not change at all within the range of capacities being considered. The high limit should be reasonable and adequate for the foreseeable future. A specific file design can be chosen on the basis of other parameters.

2 There is a significant *jump* of the cost function in the range of interest. One can attempt to design the file system to remain at the low side of the jump. Careful evaluation is needed to assure that in the worst case the storage D_{high} will never exceed the capacity at the jump. The selection of file-design choices will be constrained. The alternative is to play it safe, acquire the higher-capacity device, and use the excess capacity to increase performance. Many file-organization methods, as B-trees and direct files, respond well to increased space.

3 Many jumps appear within the range being considered. This situation is frequently true for a large database system. This means that file cost is incremental, and that there is only an economic storage constraint in choosing a file method in addition to the performance requirements. We will discuss in Sec. 5-4-6 the estimation of the incremental cost of entire systems.

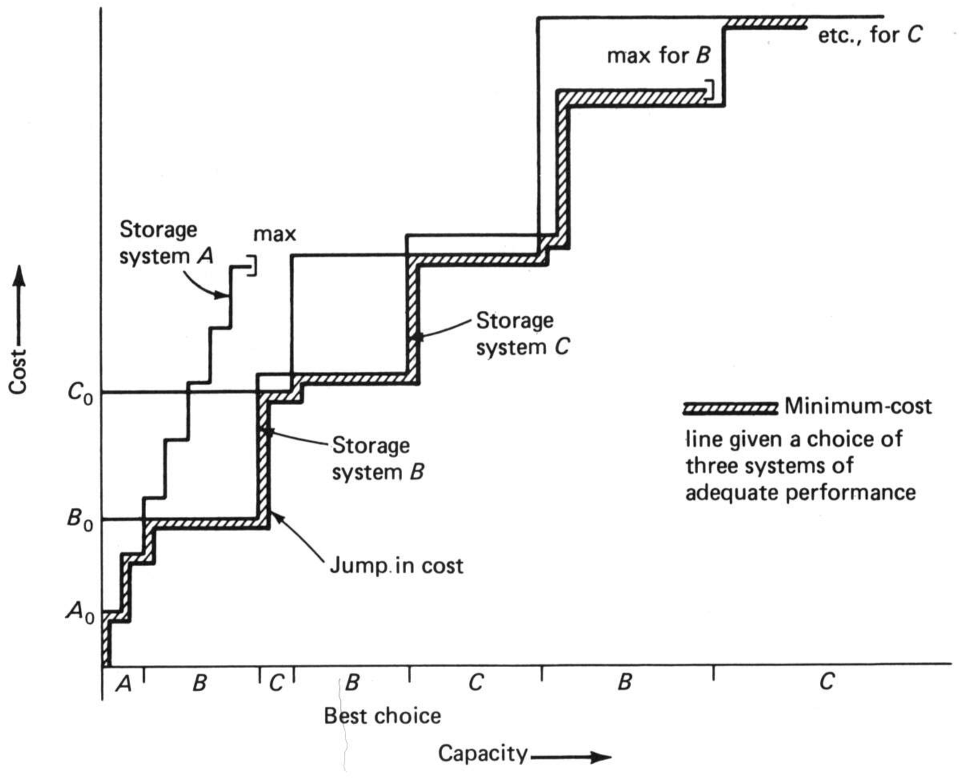


Figure 5-6 Cost versus storage capacity.

Costs are also incremental if the database operation is added to an existing computer system. The cost per increment of file storage is then known, but not controllable. The cost function starts at some point and curve of Fig. 5-6.

The method used to select the storage required may not be as mechanical as outlined above. Long-term growth expectations may make the eventual quantity D_{high}, and hence the cost range, impractically large. The rapid development of new hardware may make a design based on current storage device information impractical. If projections of future capabilties are being used, the delay between the inroduction of new technology and its availability with full software support has to be considered; three years has been a typical interval.

If one of the design objectives is a good understanding of a number of system-design alternatives, a fixed-storage-cost assumption (**1** or **2**) is misleading. In most instances, approximate methods provide adequate initial results. A reverification is always in order before a final commitment is made.

5-4 ACCESS LOAD AND CAPABILITY OF A FILE SYSTEM

Based on the known load and the file performance parameters we wish to select a system of adequate capability at a reasonable cost. We can estimate the file performance for any design, but the number of design choices is overwhelming.

One approach is to build a model of all possible file designs, and select an optimum. Methods are available which, for a constrained set of choices, will compute the optimum solution, taking one file at a time; see Sec.9-3-3.

For instance, if indexed files are to be used, we can determine which attributes should be indexed. We can also optimize space-time trade-offs for direct files. Given a number of assumptions, we can show that in a database a global optimum can by achieved while optimizing the individual files separately, one by one. In distributed systems the optimal assignment of files or parts of files can also be determined. References to a number of such models can be found at the end of this chapter.

Unfortunately, all these methods have to make many assumptions about the available alternatives, so that their model tends to be simpler than the real world. Many of them are also mathematically complex. Their use will provide an excellent starting point for further determinations and will provide insight into complex design choices.

The second approach to select files follows conventional engineering practice. We assume here either that the choices are practically limited or that a prior modeling analysis has narrowed down the set of alternatives. The process begins with the selection of candidate hardware systems which can deal with the storage requirements estimated earlier. These are then compared and scaled up or down according to the degree of over- or underutilization which we perceive during the analysis.

It is not unusual in database work that the smallest computer adequate to store the data quantity and support the file programs is adequate to perform the remaining tasks. On the other hand, it does happen that there is no single computer capable of dealing with the data flow between core storage and disk which might be required in an active database environment. Then the tasks will have to be distributed over multiple processors.

After an approximate system is selected, a file design is chosen which matches the expected utilization pattern and can be supported using vendor and in-house resources. The procedure to establish the adequacy of the selected computer system is iterative:

1 The demand on the system is estimated.

2 The available resources of the system are similarly estimated.

3 The two are compared to see if a *satisfactory match* is obtained. A match is satisfactory when the demand is less than the capability of the resources by a ratio which is a function of the precision used to make the estimates.

4 If the demand is not comfortably within the capability of the resources, then both the demand and the resources are reestimated more precisely, and a new comparison (**3**) is made.

5 If there is a major mismatch, the equipment selected, the design of the files, or the tasks to be performed are changed and the design process is restarted at **2** or **1**.

In order to simplify the exposition, we will begin with the assumption that the application using the file system will be the only use of the computer. Consideration of the effect of multiprogramming will appear in Sec. 5-4-3.

The measurement used to appraise the demand and the capability of the system is *time*. We will not worry in the presentation about the unit to be employed; in practice, it seems reasonable to scale all values for on-line systems to minutes.

5-4-1 Aggregate System Demand

The performance parameters, which were derived from the file design, provide estimates of the time required to complete the various transactions. These are multiplied by the estimates of the transaction loads to obtain the gross aggregate demand. Table 5-1 provided the rules for matching load parameters to systems. The performance parameters used here assume that this has been done.

We will perform the initial demand calculation for a period of high demand. If the daytime activities consist of searches and updates, and insertion of new data and exhaustive searches are delegated to evening operation, then the desired production during the daytime hours is computable as the sum of the activity times for all transactions to be carried out.

$$q_{total} = L_F T_F + L_N T_N + L_U T_U + L_Z T_Z \qquad \text{5-2}$$

The appropriate equivalence for T_Z is found in Table 5-1.

The total available resource Q is, at this level of analysis, simply the available time during this period, perhaps 8 hours.

To compare the values of demand and resources, we will define a utilization ratio,

$$u_{overall} = \frac{q_{total}}{Q} \qquad \text{5-3}$$

If this value is very low ($u < 0.05$), representing perhaps only a fraction of 1 hour during the day on a dedicated computer system, then we can expect few problems, and the analysis provided in Secs. 5-4-2, 5-4-3, and 5-4-4 can be omitted.

Other, highly intensive periods and the effect of other tasks should still be investigated.

In a bank the lunch hour and the time before closing may still give problems; in a hospital a busy time occurs when the physicians make their rounds prior to opening their offices. The expected evening load should also be estimated and compared.

If the value of the overall u is low, but not negligible during some periods (maybe as high as 0.2), scheduling of the load can be considered to reduce the peak utilization.

For instance, routine processing of banking work received by mail can be halted during the lunch hour. If scheduling does not reduce the utilization significantly, the analysis of Sec. 5-4-2 should be carried out for the busy periods.

5-4-2 Aggregate System Capability

A computer system actually can support in a given period a number of transactions whose weighted sum q_{total} is greater than the available time. The reason is that some operations can be performed in parallel in all but the simplest computer system. Figure 5-7 sketches a simple, but busy system to demonstrate this.

Two types of overlap are possible and will be discussed. We find that at the same time different component types will be *simultaneously* active, for instance, a disk and its channels during data transfer; and some identical component types will be active in *parallel*, for instance, two disks.

In Sec. 5-4-3 we analyze the simultaneous operation of different component types. To analyze *simultaneity*, we consider overlap between dissimilar constituents of the transaction demand: seeks, rotational latencies, and block-transfer times, since the various component types participate in only some of these constituents. Overlap of computing cycles with disk operations was introduced in Sec. 2-3-2.

In Sec. 5-4-4 we analyze the effect of *parallelism* in the operation of multiple hardware components of the same type. Having multiple components permits multiple seeks, latencies, block transfers, and computations to be carried out in parallel, and this permits increased system performance.

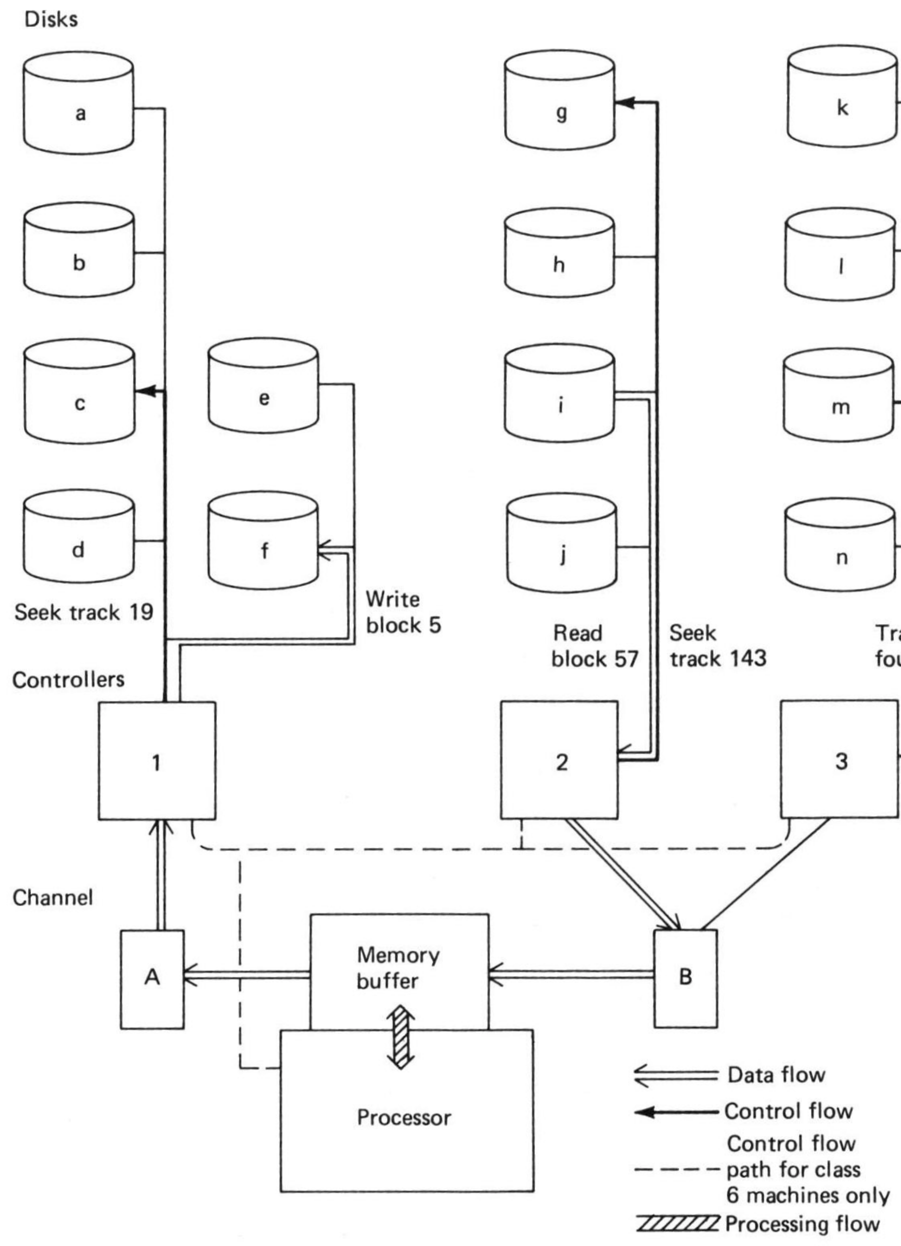
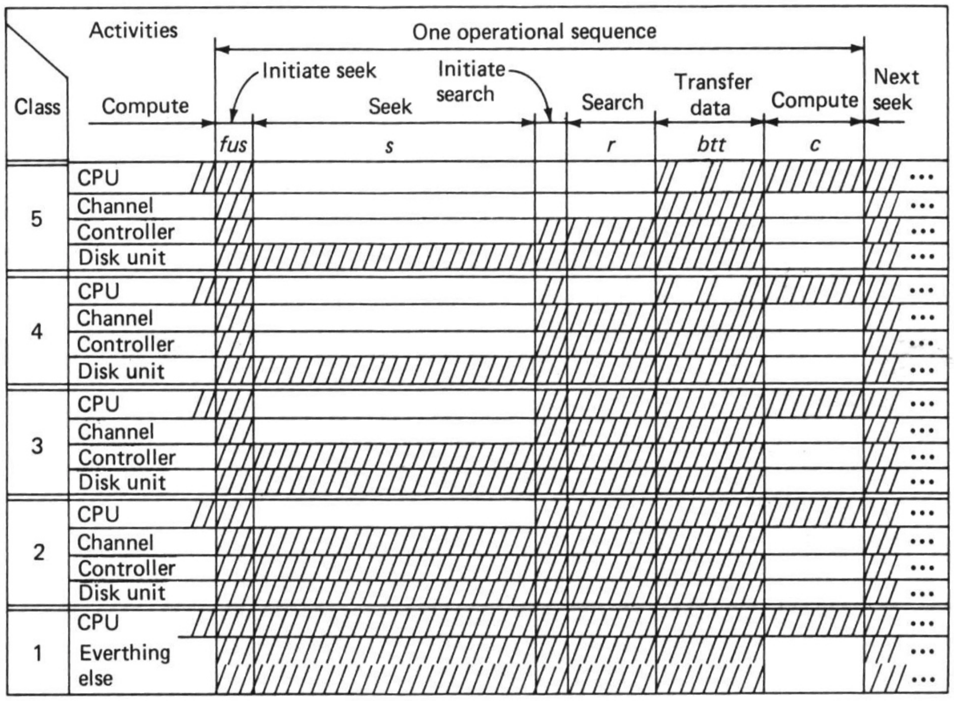


Figure 5-7 Opportunities for parallelism.

In order to overlap operation of either kind, it is necessary that the system can handle multiple processes, since individual processes are made up of sections that occur in sequential order. In the typical on-line database application there will be one computation associated with every active terminal, so that even if the computations are so simple that they are not be decomposable, there will still be considerable parallel demand. When complex information analysis is done, it becomes important that retrieval computations are decomposed into parallel processes, so that the capabilities of the equipment can be exploited.

In Fig. 5-7 data transfers are active between core memory and two disk units f and i using both channels and two controllers. At the same time seeks are taking place on disk units c and g and disk m is reporting the completion of a seek. Which combinations of operations are possible in parallel depends on the *architecture* of the storage system. Data transfer and control lines are bidirectional, so that it does not matter if the transfer activity is read or write, or if the control activity is demand or response.

Figure 5-8 defines five architectural classes to help analyze the activity of various types of storage-system components during the execution of constituents of disk operations.



Definitions: Class 5: direct seek to block
Class 4: seek to track, controller search for block
Class 3: seek to track, CPU reads to find block
Class 2: channel searches for track, CPU reads to find block
Class 1: processor searches for track and block

Figure 5-8 System components active during storage operations.

Hardware Architecture, Simultaneity, and Parallelism We consider for simultaneous operation:

1 The processor and its primary memory, characterized by the value of c.

2 The channel connecting the disks or their controllers and the processor. A channel is busy during the period of data transfer btt as well as during initiation of the seeks (s).

3 The controller which transfers data and control signals to the disks. A controller can often signal to multiple disks as it receives control requests from the processor via the channel. We also find dedicated controllers, where one controller operates exactly one disk.

4 The actual disk drive which is busy during all of s, r, and btt.

A system is apt to have multiple disk units, so that this type of device, which is used most intensely, also is the device which can most often provide parallel operation. The capability to use multiple disks in parallel is limited, however, by the type and the number of controllers and channels available and the demands placed on them for simultaneous operation. For instance, in Fig. 5-7 no other disk can be active in parallel, since a typical channel (as A,B) is active simultaneously with its attached disk (as f,i) while data is being transferred.

In order to determine the amount of simultaneity and hence parallelism possible in a disk-system design, we recognize six possible architectures of storage systems, assigning increasing performance characteristics from class 1 to class 6. Other intermediate combinations are possible and can be analyzed similarly. Figure 5-8 sketches these architectural options. The architecture class determines the hardware components active during the several constituents (seek, block search, data transfer, and computation) of an operational sequence.

In the least powerful systems (*class 1*) the processor itself, the channel (if any), the controller (if any), and the disk are fully occupied during any disk operation from the beginning of the seek until the last byte has been transmitted. The processor may be less than fully involved in slightly more sophisticated machines, but its functions may be restricted by frequent control or data-transfer demands. Most microprocessors fall into this class, although microprocessor systems may employ multiple processors and dedicate some to the tasks identified with controllers and channels. Such a system may then be assigned to one of the higher classes.

If the processor only initiates the seek, and the channel, with the controller, executes the seek, then opportunities for simultaneous operation of more than one channel become possible. In such a *class 2* system, however, the processor is required to locate the appropriate block, and hence is tied up during the rotational latency time. If the file system designed for some application often reads blocks sequentially, without seeks, the computer time available will be small, which in turn limits the benefits of multiple channels.

If the channel is not required during the seek (*class 3*), control messages can be transmitted between the CPU and the controller, so that multiple controllers can be started and seek simultaneously.

In a *class 4* system, the channel, after being instructed by the CPU, can search for the proper block and then transmit data to memory. During data transfer the CPU is slowed only somewhat by memory-access interference. In this type of system, the channel, however, is fully tied up during the rotational latency and transfer time.

If the controller accepts combined track and block addresses (*class 5*), then the use of the channel takes place only at the initiation of the combined seek and block search sequence and during transmission of data. We assume that systems of this complexity use *cache memories* for active computations so that memory access interference is negligible.

We reserve *class 6* for systems having direct communication between controllers and CPU, which will allow seek operations to be initiated even when the channel is busy. The importance of this alternative will be discussed later.

In multiprocessor systems each of the machines may fall into one of these classes. Only one active processor will carry out the CPU tasks for one request. Memory interference may occur if memory is shared among processors. Other aspects of computer system architecture also affect performance but are not captured by this classification.

5-4-3 Estimating Demand by Constituent

In order to evaluate the effect of simultaneity, we have to make an evaluation separately for each constituent: seek s, latency r, block transfer btt, and computation c in each of the terms T_F, T_N, etc. This will provide a measure of the demand for each component type. In the previous derivations of the performance measures we have actually been careful to express the parameters T correctly in terms of their constituents, so that we now can safely dissect the earlier results. We will use here the symbols s, r, btt, and c also as subscripts to distinguish the various aspects of the demand parameter q.

For the seek load

$$q_s = L_F s(T_F) + L_N s(T_N) + L_U s(T_U) + \cdots \qquad 5\text{-}4$$

or in a more general notation

$$q_s = \sum_{h1} L_{h1} s(T_{h1}) \qquad \text{for} \qquad h1 = F, N, U, Z, \text{ etc.} \qquad 5\text{-}5$$

where $s(T_{h1})$ denotes the seek component of the performance parameter.

Similarly, we can evaluate the total time spent for the rotational delay constituent, for the data transfer constituent, and for the processing constituent, so that for any constituent

$$0q_{h2} = \sum_{h1} L_{h1} h2(T_{h1}) \qquad \text{for} \qquad 0h1 = F, N, U, Z, \text{etc.}$$
$$\text{and} \qquad h2 = s, r, btt, c \qquad 5\text{-}6$$

Here $h2(T_{h1})$ denotes the constituent $h2$ of the operation T_{h1}. The values q_{h2} give the demand on the system for the constituent due to the load imposed in a given period.

We have not evaluated the processor component c throughout, but it is useful to carry it here in symbolic form to verify the assumptions made in Sec. 2-3-2 relative to buffering and the bulk transfer rate t'. If the final system requires multiple components to satisfy the load, the demand on c grows proportionally, so that faster or multiple processors will be needed.

In Sec. 5-4-4 we investigate opportunities of parallel operations of various system components, in order to determine the quantity of basic operations that a computer system can deliver.

Utilization of System Components The constituents of production demand q will have to be distributed onto the components of the system in order to obtain estimates of the utilization u for the various types of components. We refer to Fig. 5-8 to obtain the conditions for the assignment of q, appropriate for the storage-system architecture. We will assume now that we have a class 5 or 6 machine. These machines provide the greatest capability for overlap of constituents. The average utilization for a disk includes all but the computation time so that

$$u_{disks} = \frac{(1 + \mathit{fus})q_s + q_r + q_{btt}}{Q} \qquad 5\text{-}7$$

where *fus* denotes the fraction of time in terms of s, $\ll 1$ (shown in Fig. 5-8), needed to get the seek mechanism started. We will discuss this term in the next paragraph. The controller and channel can be evaluated similarly: the controller is free while the disk unit is seeking; the channel is active only during command and data transfer

$$u_{controllers} = \frac{\mathit{fus}\, q_s + q_r + q_{btt}}{Q}$$
$$u_{channels} = \frac{\mathit{fus}'\, q_s + q_{btt}}{Q} \qquad 5\text{-}8$$

If $s \gg r$, the controller can control multiple disks simultaneously without creating a bottleneck. If $s + r \gg btt$, the channel can control multiple controllers with active disks simultaneously without creating a bottleneck. The term fus' will take into account the control traffic when $s + r \not\gg btt$. For the processors

$$u_{processors} = \frac{q_c + \mathit{fus}\, q_s}{Q} \qquad \langle\text{computational load}\rangle\ 5\text{-}9$$

If we do not have accurate data on processor usage, we go back to our assumptions about buffering, where we stated that we expect that $c\,\mathit{Bfr} < \mathit{btt}$ (Eq. 2-22), so that we can state initially

$$u_{processors} < \frac{q_{btt}}{Q} \qquad \langle\text{buffering constraint}\rangle\ 5\text{-}10$$

Effect of Interference between Control-Information Transfer and Data Transfer The factor *fus* above accounts for the time it takes to transfer the control information from the processor via the channel to the disk unit. Control information is required to initiate a seek, notify the processor, or perform address or key verification on cylinders and blocks.

In a class 6 machine, there is a data-independent path for such control information via each channel to the controllers. The dotted line in Fig. 5-7 indicates this control flow path. All that is required here is the transfer of a few, (*pc*), pointers of size P, and if the same transfer rate applies to such control information as applies to data, then for each seek

$$\mathit{fus}' = \mathit{fus} = \frac{pc\,P}{t} = btt\,\frac{pc\,P}{B} \ll btt \approx 0 \qquad \langle\text{class 6}\rangle\ 5\text{-}11$$

In the other architectural classes, control information has to share the path used to transfer data, and hence the seek requests will be blocked when the channel is busy with data transfers. We consider that

1 The seek frequency is q_s/s.

2 The channels are busy with data transfer q_{btt}/Q of the available time.

3 The average delay when encountering a data transfer activity is $\frac{1}{2}btt$.

Then, in a period Q, the sum of the delays encountered when attempting to move seek commands to a controller or disk is the product of these three factors. To obtain fus' in terms of the seek load q_s, as fus is used in Eq. 5-8, the delays and the base load from Eq. 5-11 give a factor

$$fus' = \frac{q_s}{s}\frac{q_{btt}}{Q}\tfrac{1}{2}btt\frac{1}{q_s} + fus = \frac{btt}{2s}\frac{q_{btt}}{Q} + pc\frac{P}{t} \qquad \text{5-12}$$

With large values of *btt*, i.e., large blocks, fast seeks, and active channels, this factor becomes quite large, perhaps adding 25% or more to the seek delays. We find that large block sizes can have significant detrimental effects on the total channel capacity in class 3, 4, or 5 machines. This secondary demand on channel capacity is frequently unjustifiably neglected, and has surprised people who expected better file performance from their systems.

An extreme case is the IBM implementation of ISAM on a 360-type computer. The entire process of following the overflow chain is carried out by a loop in the channel-control program. This reduces the load on the CPU but makes any shared access to other devices on the same channel during this period impossible. Multiplexing capability has been introduced into the channel on the 370 series machines to mitigate this effect.

In this section we have distributed the constituents of the demand over the appropriate hardware-component categories: disks, controllers, channels, and processors. We still can expect to find that the disk component is as heavily loaded as the estimate on the entire system originally indicated, since the disk units are required for all constituents of file activity. We have not yet determined the extent of the beneficial effect of multiple disks, channels, or processors.

5-4-4 Load Distribution among Parallel Components

Having *parallel components*, that is, multiple units of the same type, permits sharing of the load assigned to the type of device. For instance, if there are two channels, each channel can be used to transmit data up to its own capacity, and it appears that the system's data-transmission capability could be doubled. If there are four disk units, some of them can be seeking and others transmitting data, again greatly increasing the aggregate capability of the system.

Two factors have to be considered when an evaluation of the increase of system performance due to parallel components is to be made

1 The design of the applications has to distribute the service demands equally over the parallel components.

2 The interference between simultaneous requests has to be assessed.

We will deal with both issues. The problem of interference is the one which can be formally addressed. The initial application load distribution requires much judgment.

Application Load Distribution The distribution of the load over parallel storage components has to balance the data quantity and the access rates to the data. Load balancing takes place at several levels.

Parallel user activity Users who work during the same time period can have their private files assigned to distinct devices.

Parallel file activity Files used simultaneously by one application can be assigned to distinct devices. This technique is common in sort programs.

Distribution of file fragments Fragments of a file, for instance, multiple indexes or pointer lists, can be placed on distinct devices, and the computation which accesses these subfiles can initiate multiple processes to permit parallel utilization.

The discussion which follows will treat the assignment problem in the aggregate sense, and the result can be applied to any level above.

If there is no access load problem, the data files may simply be distributed so that the storage space is well utilized. For *dev* parallel devices, the objective is to

$$ud_i \approx ud_j \qquad \text{and} \qquad D = \sum_i ud_i \qquad \text{for} \qquad i,j = 1,\ldots,dev \qquad 5\text{-}13$$

where ud_k is the data storage volume assigned to component number k of the component type being considered. The total storage requirement D is estimated as shown in Eq. 5-1.

If the access utilization u_{comp} of any system component of type $comp = \{disks, controllers, channels, processors\}$ is high, the designer will have to consider first of all the distribution of the access load. If the *dev* parallel devices $comp_i,\quad i = 1,\ldots,dev$ are of equal performance, the distribution will attempt to make uc_i equal.

$$uc_i \approx uc_j \qquad \text{and} \qquad u_{comp} = \sum_i uc_i \qquad \text{for} \qquad i,j = 1,\ldots,dev \qquad 5\text{-}14$$

To achieve this objective in practice, the files which receive a higher access load are placed on distinct components, and files which are less critical are distributed subsequently according to the storage capability left on the devices. Optimal placement of files within a disk unit is considered in Sec. 6-3-2.

In systems having parallel components of dissimilar performance (perhaps there are some fast disks and some are larger but slower), the rule of Eq. 5-14 is modified to consider a differential performance factor pf, now

$$uc_i/pf_i \;\approx\; uc_j/pf_j \qquad 5\text{-}15$$

since the faster device ($pf_i > pf_{average}$) can handle a larger fraction of the load. This rule could be further refined to consider the actual constituent $h2$ (see Eq. 5-6) leading to the difference in performance among parallel components.

In practice, application loads are not known precisely in advance. The optimal distribution during one period will be different from the optimum a few hours or a few minutes later, and uncertainty about the time patterns can further upset the design of a system. Assignment of loads for optimal usage may hence be deferred to a time when the operation of a database has achieved a level of demand and stability so that measurements can be made to relate the load, demand, and utilization factors L_{h1}, q_{h2}, and u_{comp}.

Increased loads may also lead to the acquisition of additional disks, channels, and processors. It might even be desirable to get more disk units than are needed for data storage in order to spread the load among more devices.

The most dramatic solution to inadequate performance is an increase of the number of processors in the system. The change from one to multiple processors may involve major changes in the operating system, and its evaluation requires a separate assessment. Once several processors are being used in parallel, further increments of processors should have no deep repercussions and can be considered as any increase in component count.

When more devices are added to a system a redistribution of the utilization is always in order. It is important that file and database systems provide the flexibility to carry out the reassignments required to keep a database operation at a near optimal level.

If no satisfactory balance of utilization can be found, *replication* of files or file fragments may be called for. This will of course increase the storage demand and the update requirements, since now multiple copies have to be kept up to date. With the increase of storage devices come typically additional controllers, channels, and possibly processors. As indicated earlier, there is a strong interaction between replication and having multiple processors. Replication becomes easier but is also often needed to keep the demand on communication among processors within bounds.

The extent of benefits from load distribution is not easy to predict. There may also be many other constraints on channel assignments, storage assignments of file-organization components, separation of indexes, and so forth, which work against optimal distribution.

Interference of Parallel Loads Even after the utilization over similar components has been explicitly balanced, we find that at any instant of time the distribution of demand is such that on some components $comp_i$ multiple requests are queued, while other components remain idle.

The extent of parallel utilization is limited by the number of simultaneous processes pr which request services from the components. In the worst active case $pr = 1$ and only one of the parallel components will actually be active. No parallel operations are enabled, even though parallel components are available.

When a large number of processes generates requests, the probability of simultaneous usage of parallel components increases, but even here no even distribution will be obtained, since the requests from the processes for the device will interfere with each other.

The actual operational sequences on this level of detail are difficult to analyze. Very simple models have, however, produced results which match the observed performance of simultaneous access to parallel components very well.

We assume that the processes request services from one of the *dev* components at random. All components have an identical performance ($pf_i = 1$). The request intervals are regular. Let us consider initially $pr = 2$ processes (P,Q) and $dev = 2$ devices (A,B). Four (dev^{pr}) combinations are possible and each of the 4 cases is equally likely

a: (P→A,Q→A), **b:** (P→A,Q→B), **c:** (P→B,Q→A), **d:** (P→B,Q→B).

We do not have to consider here which device receives the most requests. We are concerned with the delays occurring because a request cannot be satisfied while a parallel request needs the same device, and with a complementary issue: a device remains idle whenever none of the requests addresses it.

We find in the example above that in 2 out of the 4 cases (**b,c**) the distribution is good (both devices are being used) and in 2 cases (**a,d**) there is interference. During interference no parallelism occurs, and 1 of the 2 devices will be idle. The fraction of idle devices in these 4 cases is $fr_{idle} = (2{\cdot}1)/(4{\cdot}2) = 25\%$.

This interference reduces the productivity of the two disks by $fr_{idle} = 25\%$ to $Pc = 75\%$. The productivity of a single disk A would have been 100% since it would have been in continuous use. The parallel operation of the second disk B only increased the system capacity from $1 \cdot 100\%$ to $2 \cdot 75\% = 150\%$, i.e., by 50%.

If the number of processes P, Q, S,... is increased the utilization of the devices will increase. However, each process which cannot receive service will be delayed. In the example above two processes were delayed in the four cases. The average number of delayed processes is $pr_{del} = pr - Pc \cdot dev$. These delayed processes have to enter a queue. The relationship of queuing and delay is given in Eq. 6-28.

If the number of devices is large relative to the number of processes, interference will be reduced, but the devices will remain idle more often.

To estimate the productivity *Pc* of a system with parallel components a number of assumptions have to be made, and different models will affect the result. We assume here that a process, after it has had its turn, will immediately make another request and that new request is independent from the previous request. These conditions lead to a stable request pattern. More complex patterns have been explored for similar problems [Baskett76] or may be approached by simulation.

The values for the productivity *Pc* in Table 5-3 were derived by case analysis for the small parameters and by binomial approximations for the larger values. The values for the expected number of delayed processes pr_{del} were computed as above, $= pr - Pc \cdot dev$.

Table 5-3 Productivity of Systems Having *dev* Parallel Components

	Number of active processes *pr*													
	1		2		3		4		6		8		∞	
dev	P_c	pr_{del}	P_c	pr_{del}	P_c	pr_{del}	P_c	pr_{del}	P_c	pr_{del}	P_c	pr_{del}	P_c	pr_{del}
1	1.00	0	1.00	1.00	1.00	2.00	1.00	3.00	1.00	5.00	1.00	7.00	1.00	∞
2	0.50	0	0.75	0.50	0.87	1.26	0.94	2.12	0.98	4.04	0.99	6.02	1.00	∞
3	0.33	0	0.56	0.32	0.70	0.90	0.80	1.60	0.91	3.27	0.96	5.12	1.00	∞
4	0.25	0	0.44	0.24	0.58	0.68	0.71	1.16	0.85	2.60	0.91	4.36	1.00	∞
8	0.12	0	0.22	0.22	0.29	0.66	0.37	1.04	0.55	1.60	0.63	2.96	1.00	∞
∞	0	0	0	0	0	0	0	0	0	0	0	0	0.5	∞

We see in the table that the number of active processes *pr* should exceed the number of devices *dev* if systems with parallel components are to be more than 75% productive, but also that the number of delayed processes becomes significant. A high productivity becomes feasible only if the delays are tolerable, and for most on-line database operations they will not be.

To increase the productivity when $pr \approx dev$, queuing and buffers can be used to distribute poor short term request patterns over a longer time. However, when reading records from files randomly only one buffer can be effectively used by one read process. When writing blocks, multiple buffers can be employed and a good load distribution can be achieved. If some transactions can tolerate larger delays, a priority scheme may help.

Expected Distribution of Load We have given above estimates for the productivity of of similar components when they are used in parallel. The model used here assumed a uniform demand pattern of the processes to the devices.

If the load per device LpD has not been explicitly balanced, the components will not receive an equal load from the processes. It is possible to analyze the expected load distribution based on a random behavior. Of interest is here the short term load distribution; the long term distribution, given uniform random behavior, will be equal for all devices.

Some values for $LpD(i/dev)$ for two and three devices are presented in Fig. 5-9, with $i = 1$ denoting the active device at any instant. The values appear to stabilize around (0.58, 0.42) and (0.50, 0.33, 0.16). The ideal values would of course have been $1/dev$ or (0.5, 0.5) and (0.33, 0.33, 0.33). These values can in turn be used to estimate a maximum utilization factor $umax(dev)$, which will limit the eventual productivity. Since the progress of transactions computation is limited by the busiest device, we find, for the limiting cases in Fig. 5-9 (dev =2 and 3 parallel devices), that

$$umax(2) = \frac{1/dev}{\lim_{pr\to\infty} LpD(1/2)} = \tfrac{0.50}{0.58} = 0.86$$
$$\text{and} \qquad umax(3) = \frac{1/dev}{\lim_{pr\to\infty} LpD(1/3)} = \tfrac{0.33}{0.50} = 0.66 \qquad \text{5-16}$$

It is better to measure the imbalance of a system to find *umax* rather than use Eq. 5-16, since the statistical assumptions necessary for this or other analytical results can be quite unrealistic. File-access counts may be obtained by installing utilization-monitoring devices on the computer hardware. Other measurements can be obtained as part of operational statistics of the computer accounting system.

For a new system measurements cannot be taken and the estimates given above may be used as an initial design point. When hardware or its usage is unusual, simulation of the system may be required. If a reasonable facsimile of the system exists, some simple measurement programs can be used to obtain values of the maximum activity. These techniques are particularly important in a multiprogrammed environment.

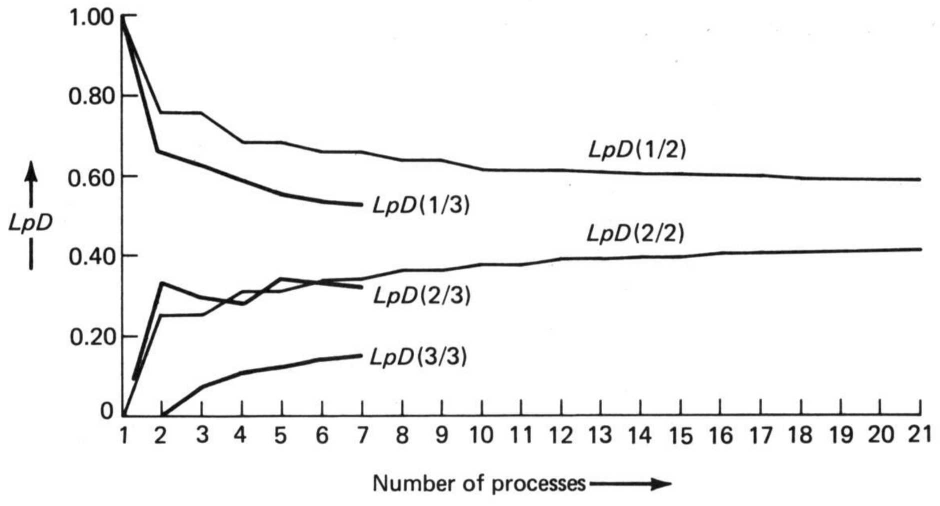


Figure 5-9 Random load distribution.

Load distributions seen in practice match approximately the pattern of Lp_D shown above if known and obvious imbalances are accounted for. It may, for instance, be wise to exclude logging or archival devices, and their assigned load. Figure 5-10 shows a long-term measurement of a file system which had actually been balanced. The test was designed so that $c \ll r, s$, so that the condition that the processes have high request rates was true. The imperfect long-term distribution may not affect the short-time distribution greatly since it is likely that at busy periods the files were used in a randomly balanced pattern. The fact that a component does not carry its "fair" share all the time cannot be helped unless the system can be scheduled in a rigorous lockstep fashion.

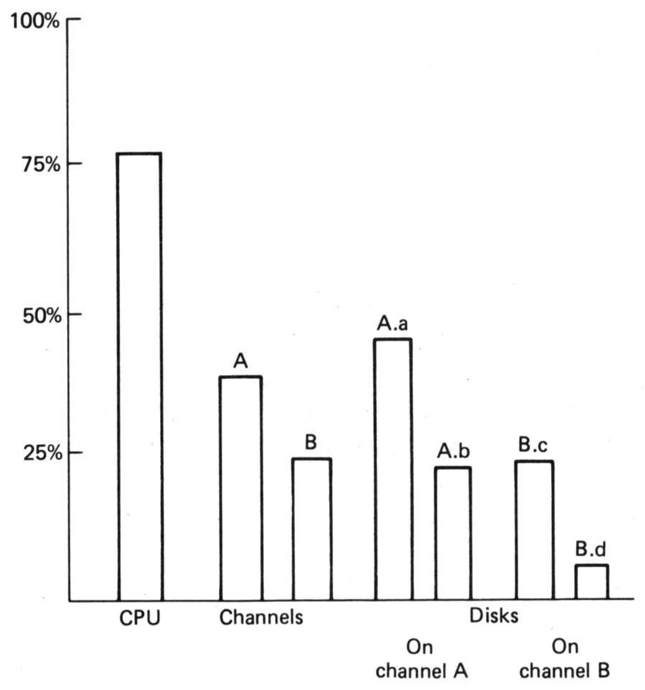


Figure 5-10 System measurement: utilization of components (data from an IBM VS evaluation).

Knowledge of disk, channel, and processor utilization based on machines using similar hardware and software architecture can provide further guidelines for prediction of realizable exploitation levels. If load imbalance, interference, or generally low productivity of a system is an issue of concern, operational data should be obtained for analysis.

It is obvious that adding more devices to a system does not increase the system capability in a linear fashion. it is, however, desirable to be able to have many processes available in order to utilize the available devices effectively.

We indicated earlier that queuing does increase the response time to requests. Techniques to evaluate the effect of queuing and also the effect of more complex service request distributions on file systems are presented in Chap. 6.

Application of Interference Effects The productivity factors from interference can be used to correct the utilization of the multiple disks, controller, channel, and processor components *comp* initially estimated in Eq. 5-14. We first apply the productivity factor to all component types having multiple ($dev(comp)$) units.

$$up_{comp(i)} = \frac{uc_i}{Pc(pr, dev(comp))} \qquad \text{for} \qquad i = 1, \ldots, dev(comp) \qquad 5\text{-}17$$

Given the prediction of the imbalance of the load as distributed to the components, it is now possible to state conditions for each device type which are requisite for the system to operate satisfactorily,

$$\begin{aligned} up_{disk} &< umax(dev_{disk}) \\ up_{channel} &< umax(dev_{channel}) \qquad 5\text{-}18 \\ up_{processor} &< umax(dev_{processor}) \end{aligned}$$

Here much smaller margins apply than were needed in Eq. 5-3, since now inequalities of load distribution and the observed productivity is taken into account. If these conditions are not adequately met, a change to the hardware or to the file-system design has to be made and another iteration of the analysis is required.

The critical periods evaluated above occurred during daytime processing of transactions. It is still necessary to verify that other operations can be carried out when scheduled. It would be a pity if a user cannot inquire into a system in the morning because the reorganization, scheduled for the night, did not complete in time. The performance parameter for reorganization, T_Y, can be used without further modification. If the reorganization time is excessive, alternate file designs will have to be considered. In very large systems techniques which allow incremental reorganization may have to be used.

5-4-5 Sharing of Facilities

We have assumed, in the analyses to this point, that the database operation was the only user of the hardware facilities. This is rarely the case, although there are many instances where database operations dominate. In an environment where the major use is related to the database, and especially in systems where the database transactions have priority, the effect of the other tasks will be minor. It can be quite profitable to share the processing capability of a system with other users which have lower priority, but who still will receive many computational cycles, since the typical on-line database operation has to have an excess of computational capability in order to be responsive. If the data-storage units are shared, some interference may result.

If the database function is not the major part of system operation, several assumptions made previously are not valid.

Processor Competition In a heavily shared environment, the condition that $c < btt$ for the file operations will not guarantee that the file system will process sequential data without losing rotational latency because of other computational demands on the processor. Processor usage by tasks which are active at the same time will reduce the share of the processor available to the database computation, so that conditions for continuous data transfer may not be met. This will double or triple the response time when using files as shown in Eqs. 2-22 to 2-24.

Disk Competition If there is active use of the same disk units by different computations, the probability of having to precede every read sequence with a seek s will be high. In this case, a conservative estimate would associate every block fetch with a seek operation, greatly decreasing the expected performance due to decreased locality. In particular, `Get-next` operations and exhaustive searches may be affected. The advantages of certain file designs cannot be obtained under these conditions. Where separate disk units and channels are reserved for the database function, the interference may be small. It may be profitable to reduce the degree of multiprogramming in order to achieve better utilization on specific tasks. This might actually increase the overall throughput.

Measurement A pilot system which models the various file transactions within the existing computing environment can provide at a modest effort useful performance estimates. The performance data obtained will be compared with those expected in an isolated operation, which are derived theoretically. The ratio provides us with multiprogramming factors, M_h,

$$M_h = \frac{\text{measured performance of } h}{\text{computed performance of } h} \qquad \text{5-19}$$

for each of the basic processes h which are part of the database computations. These factors may range in practice from 1 to 0.1.

Such tests do not require the existence of the entire, or even any, database. The pilot study consists of the execution of a proper sequence of random or sequential read or write operations, together with a comparable amount of CPU usage. A program to mimic the proposed operation can often be written and executed, on the actual system to be used, at the cost of a few days of effort. The availability of the system is, of course, crucial, and the test has to be made during typical and high-usage periods of the system functions with which the database will be sharing the resources.

Such a test is not equivalent to a full simulation study, where all computations, as well as the hardware, have to be parameterized and simulated.

Multiprogramming Alternatives The use of the database may itself require multiprogramming in order to serve multiple concurrent computations. This is, for instance, the case if a number of terminals are used for update or retrieval. The operation of one terminal can affect all the others.

Transaction-oriented systems are organized to minimize multiprogramming losses. The strategy is to attempt to finish current transactions before initiating new, possibly conflicting, ones. A transaction in progress may have the privilege of reserving a disk device until the task is completed or until the task frees the disk voluntarily. A transaction computation performing an extensive search has to be programmed by a responsible rather than by a competitive programmer and yield the device at regular intervals.

In multiprogramming and timesharing systems designed to support only scientific or engineering computations, no facilities may have been provided to hold disks to task completion, since their system objective is a fair distribution of capabilities rather than an overall system optimization. Here measurement or a worst case evaluation is needed to establish the multiprogramming factor.

The initial system-design choices made in Sec. 5-3 have to be adjusted in order to bring the utilization parameters into acceptable ranges, so that:

$$up_h < M_h umax(dev_h) \qquad 5\text{-}20$$

holds for all system components h.

5-4-6 Cost versus Capacity

At some point an adequate system configuration for the expected load and performance has been found. The file organization is also defined. This process will typically take several iterations.

It is now necessary to verify the validity of the design for a low and high level of load to be expected in the future. This can reveal problems as shown in Fig. 5-11. For application S the system design A is optimal; for application T, B is appropriate. At the average value of application U, design A is optimal, but it becomes prohibitive at the high point of the load range.

If we choose for application U system A, we will do well if the estimates of load are accurate or high. If the load grows, or the initial estimates were low, the conversion to another design has to be made. If this possibility is not foreseen, its occurrence will be embarrassing. If system B is chosen, the system operates at excessive cost until the load reaches the crossover point. The best alternative is to redesign the system so that effects of miscalculation or poor forecasting of the load will be neither fatal nor costly. The desired relationship had been achieved for applications S and T. The combinations of basic file methods and the selection of alternative hardware should make it possible to design a file system C equally appropriate for application U. The curves shown for file systems A and B are smooth representations of more realistic cost curves as presented in Fig. 5-6 which described storage system costs alone. This allows the cost of the system to be expressed in terms of the load placed on the system without getting bogged down in detailed discussion of hardware or implementation choices. Many of those

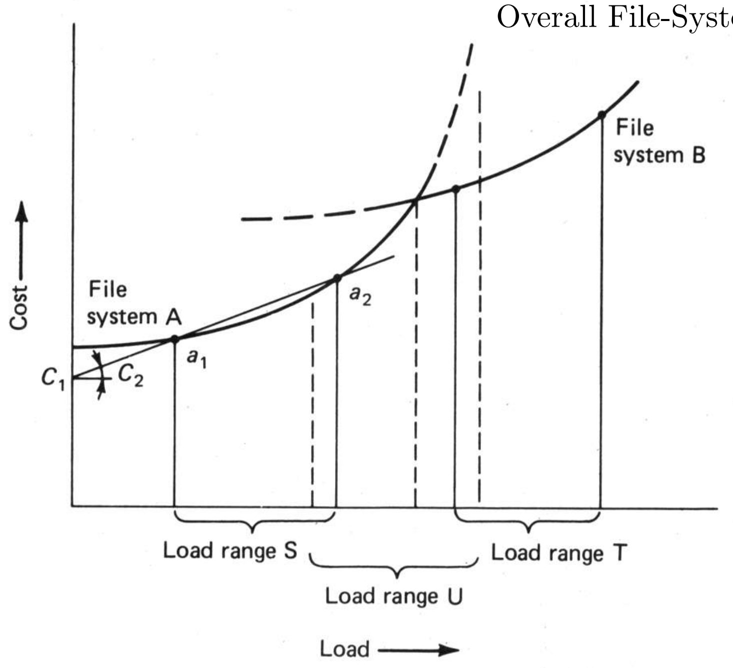


Figure 5-11 Cost versus load curves.

choices are in fact best deferred until operational experience has been gathered. The presentation of a cost versus load curve will make observers aware of incremental costs and limits, and hence is more satisfactory than such a statement as, "Another million dollars will solve our problems".

It is desirable to express incremental cost within the range of the load. Figure 5-11 shows this being done for application S by drawing a line through points a_1 and a_2. The angle at C_2 provides the rate as cost per unit load. Now the costs can be presented as

$$cost = C_1 + C_2 \cdot load \qquad \text{between } \min(L_S) \text{ and } \max(L_S) \qquad 5\text{-}21$$

While this final performance summary is not directly relevant to the system design, it is essential to provide to management the cost-to-load ratio which enables computer-acquisition decisions to be made in a manner analogous to the making of decisions in other business areas.

We have employed some greatly simplifying transformations and assumptions. Some experience is required to judge their effects in specific cases. Simplifications, however, are in common use in any situation where the need exists to provide understandable models of complex, *real-world*, relationships.

5-5 COST-BENEFIT COMPARISON

In the preceding sections, we have shown a general approach to the estimation of benefits, evaluated the storage costs, and verified that the computer system, or at least its file components, were adequate in terms of capability to handle the expected load. The latter analysis may have to be carried out repeatedly in order to achieve an acceptable solution.

We now combine the results of these sections and present them in a fashion which is related to basic business-economic terms.

5-5-1 Revenue versus Cost

A business which sells consumer goods places a price on its goods and expects to receive an income directly proportional to the amount of sales. Such a revenue curve is shown in Fig. 5-12*a*. Only a business in a monopolistic position will not be able to increase revenues by increasing sales and will have to resort to other tactics in order to boost income.

At the same time costs increase as production increases from a fixed basis at a zero production level. Some of this cost increase may be linear, that is, directly associated with quantity, as would be the cost incurred for supplies consumed in manufacturing. Other costs will increase less in proportion to quantity owing to efficiencies of mass production. The cost associated with sales will tend to increase somewhat faster with quantity since, after an initial demand is satisfied, more effort is required to convince additional customers. A sample of cost curves is sketched in Fig. 5-12*b*, and their total is presented in Fig. 5-12*c*.

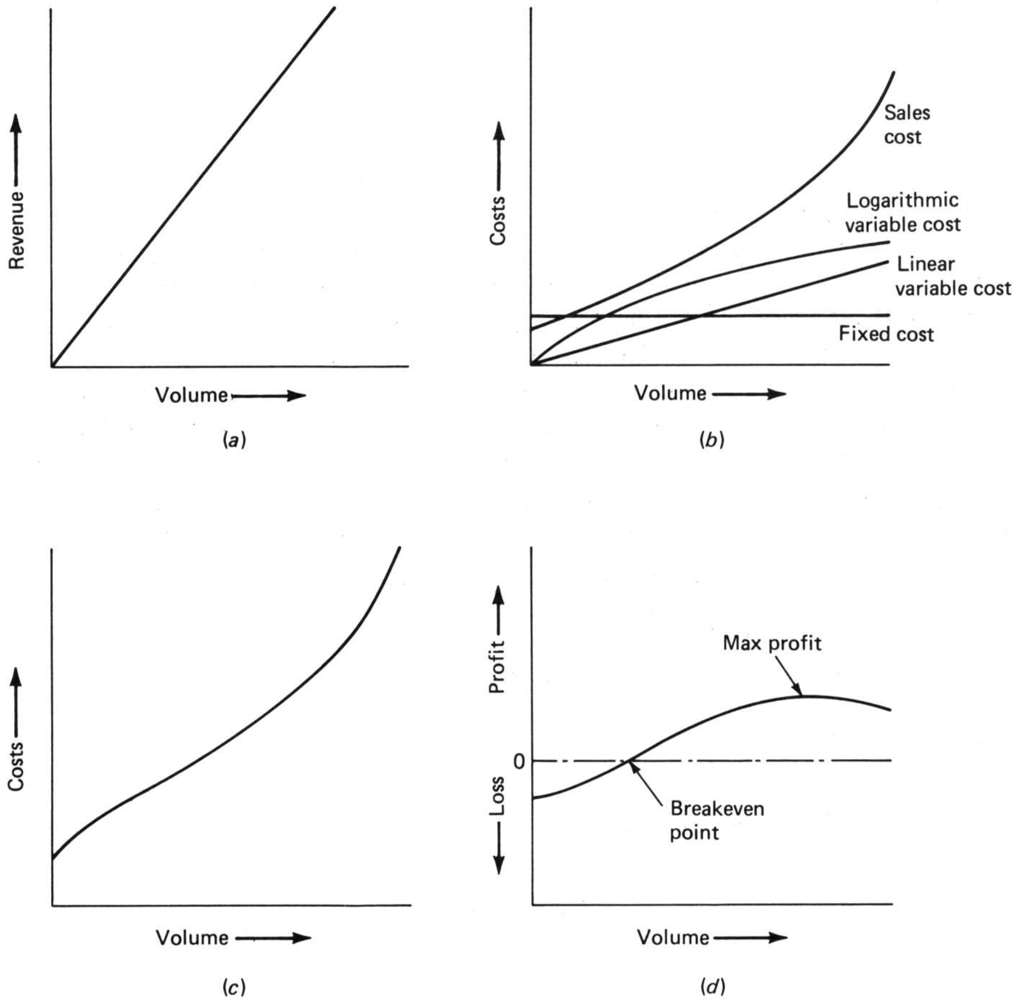


Figure 5-12 Basic manufacturing revenue and costs.

From the revenue and cost curves, a profit and loss can be derived which shows the break-even quantity, and the volume where the maximum profit as obtained (Fig. 5-12*d*).

In the information-selling business, the projection of costs and revenues is based on system capability and utilization. While the cost figures will be similar, the evaluation of benefits of an information system is more complex. An understanding of relevant factors is necessary when we wish to appraise the viability of a database system.

Benefits of Information We discussed the benefits earlier in some of the factors which control the realization from a database system. In order to relate benefits to costs, we can consider as factors the quantity of data available and the intensity of the use of the database. Figure 5-13 sketches some of these general considerations.

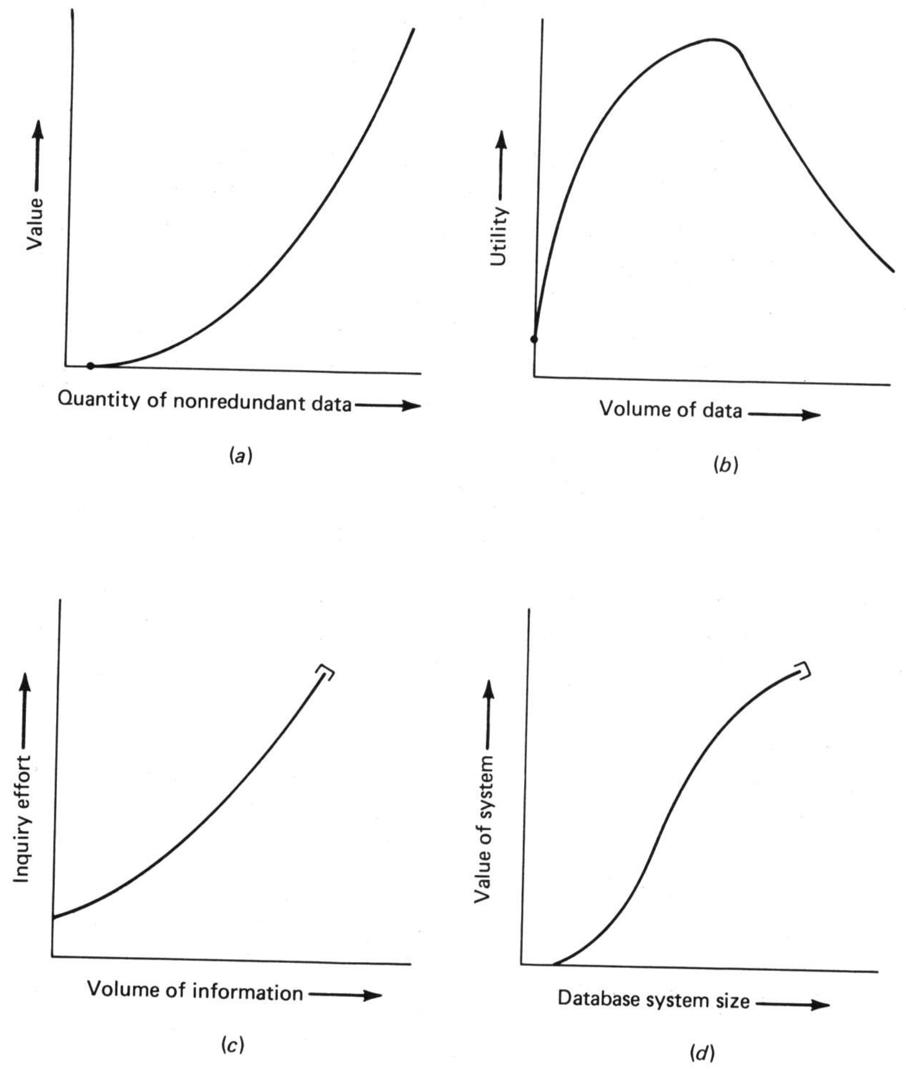


Figure 5-13 Benefits of information.

Information theory states that the information content of a nonredundant set of data increases exponentially with the quantity of data (Fig. 5-13*a*). In practice, redundancy will also increase with volume in order to maintain accessibility and this will reduce the rate of increase. The curve shown also takes into account that users have some prior knowledge, so that small data quantities have negligible information value. On the other hand, we find that as information increases, our ability to improve decisions does not keep pace. Most of the decisions made in enterprises are based on incomplete knowledge and yet appropriate to the general situation. Frequently statistics of relatively small samples can provide adequate answers with high reliability. This decrease in practical utility may have an effect as sketched in Fig. 5-13*b*.

In order to extract any information out of the database, it has to be queried and processed. Extracting more from the database requires more frequent or more complex inquiries. The effort to retrieve information will increase more steeply than the amount of applicable information obtained, as shown in Fig. 5-13*c*. The cost to process data is often proportional to $n \log n$, where n indicates the data quantity. When the value which is returned to the users diminishes to the point that it is no longer an effective application of their effort and funds, they cease to inquire further.

When all these factors are combined and displayed, relative to database-system size, the *revenue* curve may be of the form shown in Fig. 5-13*d*. The actual shape and the values will differ from application to application, but we notice the disparity between benefits from the sale of information and the sale of goods shown as Fig. 5-12*a*.

5-5-2 Costs

The cost of the storage facilities is a major component of file-system cost. There are many instances where the potential value of data is less than the minimal cost of storage. Consequently, one will want to verify that we can afford to store the data that we wish to process. In data storage there are often some economies of scale as was seen in Fig. 5-6. The marginal costs of additional storage can be obtained as Cm in Fig. 5-14, which presents the same data in smoothed form.

Charging for Storage When marginal costs are used to justify expansion or new projects, the fact that a low charge based on marginal cost hides a large basic storage expense Cd must not be forgotten. If earlier projects are charged at a higher marginal cost for data storage, unhappiness is sure to arise. If the unhappiness causes earlier projects to stop or atrophy, then the marginal cost of the later projects increases so that charges have to be increased. Now everybody is unhappy. A charging rate Cp based on strictly proportionate rates has the problems that does not reflect marginal costs well and leads to losses until the load exceeds capacity CP and to excessive profits thereafter. A desirable compromise can be a charging rate Cb which matches costs over a large fraction of the operating range. This will leave an amount Ce uncovered by storage charges, which will have to be obtained as a surcharge on database processing or data entry.

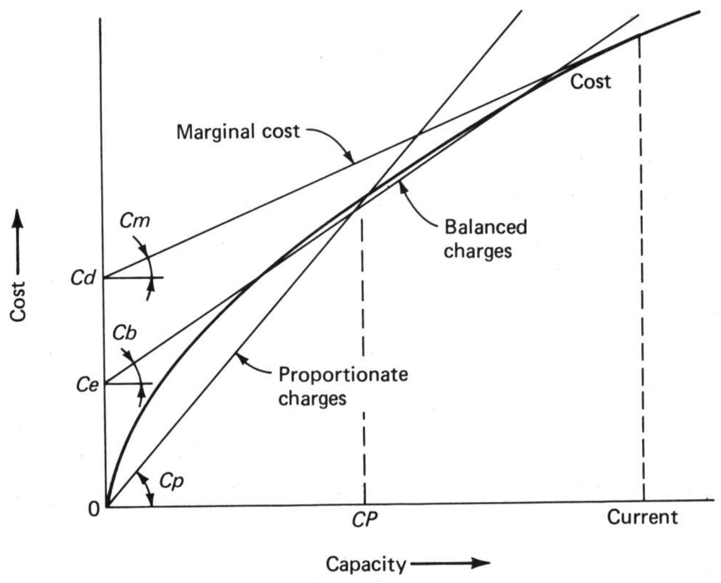


Figure 5-14 Marginal storage costs.

Companies which provide database services tend to use a balanced charging approach, since this encourages users to store data on their systems, which in turn will encourage data analysis. Companies which sell computer services tend to use the proportionate charging method, since they view file storage as an auxiliary service to the user which must pay its own way. Large databases are rarely found on such systems, although some companies have invested in database programs for their users.

Cost of Processing The processing costs for a given computer system have to be viewed in terms of performance. The addition of devices, such as channels and disks, increases the processing capability of a system, but the increase in capability is less than proportional. Increased interference, inability to distribute the load evenly, and dependence on specific critical components all combine to decrease performance beyond a certain usage volume.

Different systems may have different performance curves, but their shape will remain similar, as shown in Fig. 5-15. Systems B and C may have equal costs but differing processing and storage-arrangement algorithms, so that B is better at a relatively low load, but C outperforms B at a high load. System A would be attractive only if the cost is sufficiently less so that the performance difference does not matter.

The difference in these systems leads to different productivity. Productivity is a measure of benefits obtained taking into account that the performance must be realizable (Sec. 5-2). The result is shown in Fig. 5-15*b*, which combines the effect of increased productivity in terms of requests processed, and the negative effects of poor system performance.

Since there are just about infinitely many systems of equal cost, as B and C in the figures, then for every load a system could be chosen which would optimize productivity. The dashed line indicates the envelope describing the set of these optimal systems.

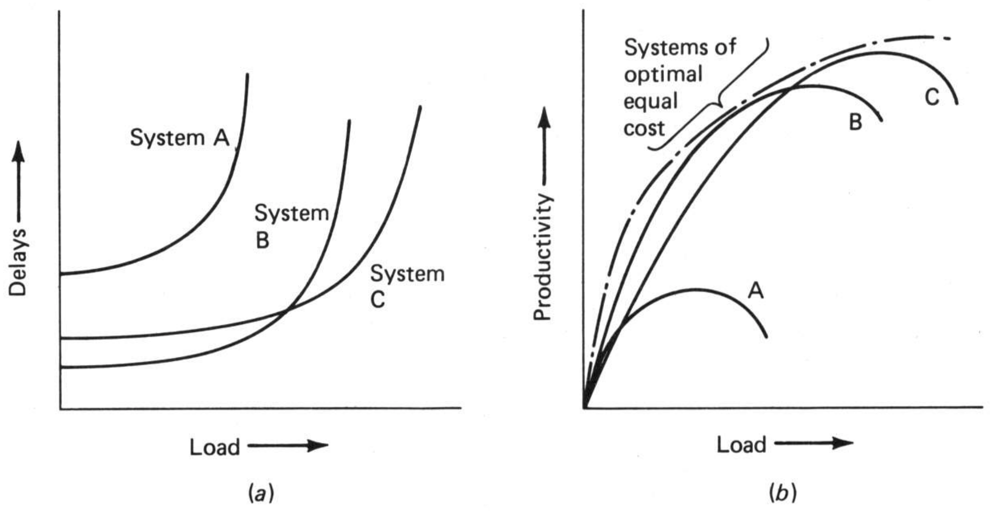


Figure 5-15 System performance versus load.

This information can also be presented on the basis of system cost for a given load. Cost-performance curves have been presented by measuring existing, rather than optimum, systems. It has been shown that the raw hardware (mainly CPU) performance increases steeply with cost. A rule of thumb, referred to as Grosch's law, has been that,

$$\text{Computer performance} = C_G \text{ cost}^2 \qquad \text{where } C_G \text{ is some constant} \tag{5-22}$$

and this rule has been verified on IBM equipment of the early 360 type using a variety of scientific problems [Solomon[66] and Knight[68]]. There is, of course, the possibility that IBM used Grosch's law to determine its equipment prices, since it must be nearly impossible to relate prices to costs for a large integrated company. For data-processing operations, however, the same machines showed

$$\text{Data-processing performance} = C_G \text{ cost}^{1.5} \tag{5-23}$$

which still indicates advantages of scale, but to a lesser extent than seen in scientific computation.

The rule is less applicable to file-oriented and interactive systems, since it does not account for interference or for loss of productivity due to poor performance at the user terminal. While it is easy to concentrate on a single aspect when analyzing computers, it is obvious that in the more modern systems, raw processing power only indirectly increases throughput. Since file capability is governed more by design decisions than by computer equipment performance alone, software and its implementation become increasingly important. Interference among multiple computations is another area not solved by more powerful hardware. No satisfactory rules are available to decide how many functions a designer should provide within one system.

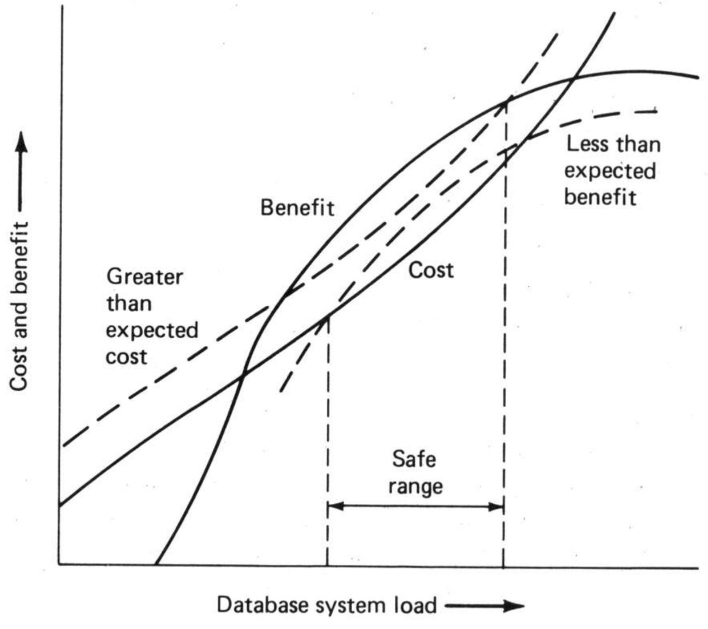


Figure 5-16 Total costs and benefits.

5-5-3 Summary

If we now combine the three elements: benefits, storage costs, and processing costs, and plot them only over the limited range of interest, we may generate cost-benefit lines as shown in Fig. 5-16.

When benefit and cost curves are similar in shape, the lines are relatively parallel. Parallel lines indicate that small errors in cost or benefit estimation will have a great effect on the range within which the system is profitable or effective. This example presents an undesirable but not unusual operation. Good documentation of the design effort and its parameters is essential, in order to be effective in manipulating the outcomes represented by these curves.

In practice, the cost picture gets even more complex because of discounts, special deals, and peculiar amortization methods. From a technical point of view, these anomalies are best disregarded and left to the financial staff to be dealt with independently of the system design. It is rare that the benefit of *good deals* exceeds the long-range cost of suboptimal system design. In order to make such issues clear, processing personnel have to be able to produce sensible and understandable numbers when they wish to present their views.

People outside of the computing clique often perceive correctly that many justifications for computer systems are only technical mumbo jumbo.

BACKGROUND AND REFERENCES

The process of system design, evaluation, and acquisition brings together people with technical, financial, and managerial orientation. In such a mixed environment clarity of expression, and measures which seem objective, are important communication tools. Graphical presentation can often help to present expectations, but can also be used to hide simplified assumptions. Brooks^{75} presents much insight and data regarding the design process. A basic reference on the economics of computers is Sharpe^{69}. Davis^{74} and DeMarco^{79} present a management-oriented approach. Davis^{82} introduces a number of papers oriented toward requirement analysis of enterprises.

Wiking^{71} and Lucas^{73} prescribe the evaluation of information systems and Lucas^{75} considers the feedback in between user benefit, usage, and performance. Alford^{77} considers real-time constraints. Sevcik^{81} presents a layered technique of load modelling. Robey^{82} defines the stages of system development. A language for expressing requirements is presented by Ross^{77}.

Human factors are cataloged by Martin^{73}; an important study is Reisner^{77}. The subject has its own journal (*Human Factors*), published by the Human Factors Society in Santa Monica, CA. Important background in this area is provided by Sterling^{74,75}. A conference on the subject was sponsored by the National Bureau of Standards (Schneider^{82}). The cost of errors is discussed by Stover^{73}. Morey^{82} considers those errors that are due to delays. Arden^{75} contains a number of papers on operating-system aspects of interaction and provides a useful bibliography.

Cost estimates used for projections change rapidly with time. At times projections are incomplete, for instance, reductions in mass-storage costs are projected (Copeland^{82}) but the studies do not consider what is happening to processing and software costs. The spread of small computers has confused the picture greatly.

Most implementation descriptions consider benefits and costs, but the data are often difficult to compare. Some references are cited in Chapters 1, 9, and 10. Batteke^{74} has details of a benefit evaluation in water management. Boyd^{63} analyzes benefits in a manufacturing system, and Loo^{71} describes the real-time needs for airlines. King^{78} provides a survey of development tradeoffs.

The structure of file-system hardware is presented by Hellerman^{73} and analyzed by Madnick^{69}. Teorey^{78} analyzes parallelism of disimilar components. Distributed systems designs were looked at by Peebles in Kerr^{75} and Morgan^{77}. Streeter^{73} found few reasons for distribution.

Formal procedures to select file structures have been developed by Schneider in ref[King:75], Hammer in Rothnie^{76}, Katz in Chen^{80}, Yao^{77}, and Whang^{82}. File-method selection can be aided by FOREM (Senko^{73}). Waters^{74} describes the decision points. Cardenas^{79} surveys file design, access structures and analyzes multi-attribute use and selection for partial-match queries. Gambino^{77} presents a model for multiring file design. Lowe^{68} and Long^{71} analyze text files.

Ramamoorthy^{70}, Arora^{71}, Parkin^{74}, Lum^{75}, and Trivedi^{81} evaluate distribution over a hierarchy of components of differing performance, and Major^{81} derives the complexity measure. Chu^{69}, Hoffer in Kerr^{75}, and Du^{82} evaluate the optimal allocation of files to multiple components. Lin^{82} considers disk allocation in a local network. Chen^{73} and Piepmeier^{75} provide optimization policies to balance the load distribution over multiple devices. Dearnley^{74M,O}, Stocker^{77}, and Kollias^{77}, as well as Hammer^{77}, have done work on automatic optimization of file allocation. Marron^{67} trades processing cost versus storage cost.

Interference problems are presented by Atwood^{72} and analyzed by Omahen in Kerr^{75}. Berelian in Merten^{77} considers the effects of paging in a multifile database. Siler^{76} and

Baskett[76] provide results of straightforward evaluation models which show the limitations of benefits obtained by distributing files over multiple devices. Measurement of database operations is described by Krinos[73], Huang[74], Sreenivasan[74], and Heacox in Kerr[75].

Miller[70] has developed decision procedures for the acquisition of large computer systems. Applications of decision-analysis techniques are presented by Easton[73]; Keith in Cuadra[70], Langefors[73], and Miyamoto in Kerr[75] have models for evaluation of information systems. King[81E] addresses general evaluation issues. Grosch's law has been tested by Solomon[66] and Knight[68]. McFadden[78] summarizes cost and benefit issues.

EXERCISES

1 a Write a proposal to install a data base system in some application area you know. Do a good *selling job*.

b Write a critique of such a proposal from the point of view of someone who will be negatively affected in some sense by the system.

c Now prepare an evaluation to management that lists the arguments for and against as fairly as possible, and then summarizes in management terms the decisions which have to be made by management in order to evaluate the value of the proposal. Assume that you are a private consultant.

2 Describe some of the costs and benefits of the *least* useful file system, including manual data collections, that you have encountered.

3 Using the manufacturers' description of a hardware file system, determine which class (Sec. 5-4-2) of file architecture it belongs to. Then evaluate the software system provided and see whether all the potential of the architecture is available.

4 An inquiry system has 3000 daily requests into a direct file of 50 000 records, each of 250 bytes. When a record is brought into core storage it has to be available for an average of 5 seconds. The system is used 8 hours per day, 300 days per year. A 2314 type disk is used for the file. Determine the minimum-cost file design. Included are the hardware cost ($200/day), the disk storage, and the buffer memory.

5 Generalize the program for Exercise 3-40 for complex questions. Such questions may involve references to multiple files as well as usage of multiple indexes. As in prior questions list any assumption made in a table to be appended to the answer to this question.

6 Derive a predictive formula which can be used to give an estimate of the length of an inquiry response time from the inquiry parameters and the file dictionary only, using again the same file design.

7 Evaluate the formula derived above for the following query:

"`Find an unmarried man between 25 and 30 years old with recent experience operating Caterpillar D-10 equipment.`"

where the operating history is given in an equipment file and the personnel data are provided in a personnel file. Use values from Table 2-1 for the parameters of the storage devices.