Modeling File System Organizations in a Local Area Network Environment[1]

Domenico Ferrari
Tzong-yu Paul Lee

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley

This paper presents a modeling methodology for the design of file system organizations in a local area network environment. We first propose a characterization for the workload of typical business transaction systems, and then use this measurable characterization to derive input parameters for the file distribution graph models and the queuing network models that represent two completely different file system organizations, i.e., a file server-based file system and a distributed file system. Total system throughput and mean system response time are the indices used in comparing these two design approaches. An example of this comparison is given. It is shown that this methodology and these models are useful tools for the evaluation of certain design tradeoffs.

Key words: Distributed File Systems; File Distribution; Queuing Network Models; Workload Characterization.

1. Introduction

Technological progress in microelectronics and data communications has made it economically feasible in recent years to build distributed computer systems connected by a computer communication network [Sch78]. The behavior of such systems can be considerably more complex than that of single-machine systems; hence, the need for design methodologies and tools is even more evident than in the latter case. We are particularly interested in the design of file systems for local-area-network based distributed systems [Stu80]. This paper focuses on a design methodology that evaluates two approaches to file system organization, i.e., a file server-based system [Mit82] and a distributed file system [Gol83]. A file server-based system has one or more dedicated machines that perform file system functions for the other hosts on the network. On the other hand, a distributed file system does not have this separate type of server : file system functions, storage and access, are distributed over all the hosts connected to the network.

The communication network in the distributed systems our study is concerned with is assumed to be a high-speed local area network [Cla78]; since all host systems are homogeneous, any user process can be executed on any host system. The workload parameters used in our models were derived from the

[1]The research reported here has been supported in part by the State of California and the NCR Corporation under MICRO Grant No. 1-532436-19900.

measurement of small business transaction-oriented systems. These systems tend to have relatively stable workloads in terms of both the behavior of each type of transaction and the number of files shared among transaction processes.

In Section 2, we propose a characterization for the workload of a typical business transaction system. Performance indices of interest are also discussed. Using this characterization, in Section 3 we distribute shared files over several given hosts in order to minimize remote disk accesses. With this distribution of files, the probabilities of remote file accesses can be calculated for each transaction type and used as inputs to queuing network models. In Section 4, details of the queuing models for the two organizations are presented : one with a dedicated file server and the other with a distributed file system. A simple but comprehensive example is given in Section 5 to show how these models can be used to evaluate design tradeoffs. In particular, the example shows that a moderate increase in CPU speed may be more cost effective than the addition of another host to a distributed system. This tradeoff can be carefully evaluated using the simple methodology presented here before a major procurement or upgrading decision is made. Avenues for future research are mentioned in the last section.

2. Workload Characterization and Performance Indices

2.1. Workload Characterization

The workload of a business transaction system can be characterized by the number of transaction types, the resource demands by each transaction type, and the relative throughputs of these transaction types during some typical interval of operation. Each transaction type uses a few temporary (private) files while sharing some permanent files with other transaction types. It is this collection of shared files that are of concern to us in this study.

The resource demands of a transaction of each type can be further characterized by the following quantities :

(1) the number of user interactions (or commands);
(2) the total CPU time demand;
(3) the number of physical disk accesses to each shared file;
(4) the number of display outputs.

Note that a display output is the transfer of a character string by a transaction process from an I/O channel, usually an asynchronous communication line, to a user terminal display device. There are generally a number of display outputs during an interaction.

These data can be collected by a suitable hardware monitor and/or by appropriate system software instrumentation. The values of the characterizing parameters for each transaction type generally remain invariant with respect to changes in the underlying structure of the distributed system; in other terms, they are usually the same for a system with a file server as for a system with a distributed file organization.

Some of this workload information is used in the next section to determine the distribution of shared files over the hosts. Input parameters to the queuing network models are also estimated from some of the quantities in this characterization.

2.2. Performance Indices

There are two specific types of performance indices of interest in this study. One is the overall system throughputs for the various transaction types, the other is the average response time to user commands for each transaction type. These two indices can be employed to compare designs of different file system organizations. The two indices may be assigned as performance requirements to be satisfied by the design. The designer must properly configure the system to meet these requirements while keeping its cost at a minimum.

Other performance indices such as the utilizations of critical resources may be useful for bottleneck detection and capacity planning.

3. Distributing Shared Files Over Hosts

Although distributing shared files over hosts is a problem which arises only in the design of a distributed file system, a similar problem is met in the design of a file server system if the distribution of files over several disk drives is considered. Before discussing the distribution of files, we need to investigate the problem of allocating transaction types to hosts.

3.1. Allocation of Transaction Types to Hosts

Since files have not been assigned to hosts yet, in this phase of the design we assume that they are stored in a single place on the network (e.g., in a file server). Hence, the only load that can be meaningfully balanced at this point is the CPU load. Balancing the I/O load is the objective of the next phase.

Given the number of hosts that the distributed system to be designed will consist of, transaction types are assigned to hosts according to their expected CPU demands per unit time, properly weighted by their relative throughputs. The CPU demand rate is obtained by dividing the total CPU demand by the uniprogramming execution time of one transaction. The uniprogramming execution time of a transaction is given by the sum of its total CPU time and total I/O time, and is used here instead of the multiprogramming execution time since this is unknown. The objective of this assignment is to balance the CPU loads among the various hosts as much as possible. The criterion we use is the minimization of the sum of squared CPU demands for each host. Specifically, if the CPU demand rate for transaction type i is d_i, and there are m transaction types to be allocated to n hosts, the objective function we choose to minimize is

$$\sum_{j=1}^{n}\left(\sum_{i=1}^{m} x_{ij} d_i\right)^2 ,$$

where the decision variable x_{ij} is 1 if transaction type i is allocated to host j, and 0 otherwise. A simple heuristic can be used to solve this generally NP-complete problem [Cha75].

Note that we generally try not to assign the same transaction type to more than one host since transactions of the same type exhibit the same file access pattern. This approach will ensure better locality of accesses to (single-copy) shared files.

3.2. A Graph-theoretic Model of File Distribution

As shown in Figure 1, we construct a (bipartite) graph model with two types of nodes; a node on the left side represents a transaction type and a node on the right side represents a shared file. The edge between a transaction type node and a shared file node is weighted by the rate of file (disk) accesses from the transaction type in question to the corresponding shared file. The rate of file accesses is obtained by dividing the number of file accesses by the uniprogramming execution time of the average transaction of that type.

Now, we can use the assignment of transaction types to hosts determined in the previous phase to coalesce some of the transaction type nodes and their associated edges. For example, if transaction types 1 and 4 are assigned to the same host, we combine nodes 1 and 4 in the graph in Figure 1. The weights of the edges from this new node to the shared file nodes are computed by summing the individual weights in the original graph model. This is shown in Figure 2. We need to retain the original graph model in order to be able to calculate remote file access probabilities on a per transaction type basis later.

Formally, let us assume that the number of hosts is n, the number of shared files is k, and the weight of the edge from node i to j is c_{ij}. The objective of the assignment of shared files to the n hosts is to cluster the hosts-files graph so that each cluster includes one and only one host, and the sum of all inter-cluster edge costs, i.e., the total rate of remote file accesses, is minimum. Mathematically, we can formulate this as a standard integer programming problem [Sal75]. The decision variable x_{ij} is 1 if file j is assigned to host i, and 0 otherwise. Notice that the sum of both inter and intra-cluster edge costs is a constant for a given graph. It is easier in this case to formulate the problem in terms of the sum of intra-cluster edge costs. The objective function to be maximized is then

$$\sum_{i=1}^{n}\sum_{j=1}^{k} c_{ij} x_{ij} \ .$$

This problem can be solved by a simple polynomial-time algorithm [Lee83]. However, if there are capacity constraints on some hosts, then we can apply a standard (binary) integer programming solution technique [Sal75]. The capacity constraints can be represented as follows :

$$\sum_{j=1}^{k} x_{ij} s_j \le L_i \text{ for } all\ i \ ,$$

where s_j is the size of file j and L_i is the capacity of host i.

After clustering the shared files, we can use the original model (see Figure 1) to compute the probabilities of local and remote file access (one for each remote host) for each transaction type.

4. Queuing Network Models of Distributed File System Organizations

In the third phase of our design methodology, we construct a queuing network model of the distributed system. This model offers a relatively fast and inexpensive way of verifying whether the given performance requirements can be satisfied by the system. We illustrate how simple queuing models of the two basic file system organizations can be constructed, and how they can be used to compare the performances of these organizations. Our models will be product-form queuing networks [Bas75], that can be analyzed quite rapidly and inexpensively by any of the several solution packages now available commercially. The hosts and the local area network will be first modeled separately, then integrated into the distributed system model. We assume that in our system each CPU is multiprogrammed, and has one disk drive and several terminals connected to it. It is also assumed that the local area network uses a multiaccess protocol.

4.1. A Model for a Host

The host model includes a number of active user terminals which initiate interactions by entering commands or data. Each interaction typically requires some CPU bursts, a few disk accesses, and some display outputs before ending by returning to the user at the terminal. Figure 3 illustrates the model of a host. The CPU is modeled as a PS (processor sharing) server, which approximates a time-sharing processor. The disk drive is modeled as a FCFS (first come first served) server, and both the display output and the user terminals are modeled as infinite servers (IS) .

A typical process will use the CPU, do a disk access or a display output, use the CPU again, then the I/O subsystem again, and so on, until it returns, after its last visit to the CPU, to the user terminal. We generally assume that the number of active terminals is fixed, and that each host can process more than one transaction type. All transactions of a given type have statistically identical behaviors, and constitute a single chain in the model. Each terminal, in our model, keeps entering transactions of the same type, but a host can have terminals corresponding to various transaction types.

As far as directories for the distributed file system are concerned, we assume that each host has a complete directory of all permanent (shared) files, and that, because of the stability of these files, directory updates are so infrequent that the network traffic and the overhead caused by them have negligible effects on performance.

4.2. A Model for the File Server

The file server is just a specialized host which has no user terminals, hence, no display outputs. In reality, a file server can achieve economies of scale by using high-density disk packs and high-speed disk channels, and distributing the cost of this equipment over all hosts. However, these considerations are beyond the scope of the models described here, which are intended to provide first-order predictions of performance. Figure 4 illustrates the model of a file server.

4.3. A Model for the Local Area Network

The backbone of the distributed system we are considering is the local area network. This component does not necessarily behave as a FCFS server; however, this approximation will generally be appropriate when the network is not heavily loaded. If we make this assumption, the local area network can simply be modeled as a FCFS server with arrivals from all hosts and departures towards them.

4.4. Distributed System Models

Models for the distributed systems being studied can be obtained by simply integrating a number of host models with a local area network model, and possibly with a file server model if required. Slight modifications of the host model are needed, however. Specifically, in a file server-based model, there is no local disk drive within each host. All disk drives are grouped together under the control of the file server's CPU. Therefore, a disk I/O operation would go through the network node to the CPU of the file server to access file server's disk drive, and back through the network node again. Disk I/O's from all hosts are essentially "remote" accesses, and they must compete for file access among themselves. Remote disk accesses in the distributed file system model are similarly routed. A remote disk access will go through the network node to the remote host's CPU first, then to the remote disk, then to the remote CPU again before coming back to the local CPU through the network node. A remote access must compete with processes running on the remote host for both the remote CPU and the remote disk drive. Note that processes will always return to their "home" host because of their belonging to the chain that corresponds to their transaction type, and of the assignment of each transaction type to a single host. Note also that, if a method were used in the first phase (see Section 3.1) which assigned transactions of the same type to more than one host, that type would have to be subdivided into subtypes, each consisting of the transactions of that type assigned to a particular host, and a distinct chain should be used to model each subtype in the third phase in order to ensure the return of each process to its home host after a remote access. Models for these two types of distributed systems are shown in Figures 5 and 6, respectively.

5. A Simple Example and Its Analysis

5.1. A Simple Example

A simple example has been chosen to illustrate the use of the queuing network models introduced in Section 4. We make the following assumptions about the models. All hosts have the same number of active terminals, and all terminals are used to input the same transaction type. In the model of a distributed file system, the probability of remote disk access by a process is the same at all hosts; furthermore, if the disk access is remote, all other hosts are equally likely to be accessed.

The characterization of the transaction type we have chosen is as follows :

(1) the number of user interactions in a transaction is 47;
(2) the total CPU time demand is 16.7 seconds;
(3) the number of physical disk accesses for all shared files is 179;
(4) the number of display outputs is 217.

We assume that each interaction of a transaction uses on the average the same amounts of resources. The CPU bursts and the branching probabilities to disk, terminal, or display server are calculated by assuming geometric distributions of the number of visits to the CPU. Other important parameters are the user think time, the disk service time, the network service time, the display output service time, and the file server's CPU service time. They were chosen to be 1 second, 32.07 milliseconds, 0.17 millisecond, 14 milliseconds, and 5 milliseconds, respectively.

Most of the above assumptions were made to reduce the number of model parameters that could be varied. Three control variables were considered in this experiment, for both the file server-based model and the distributed file system model. The total number of terminals ranged from 1 to 36, the number of hosts in the distributed system was either two or three, and the probability of remote disk access were either 0.05 or 0.30. Only numbers of terminals that are integral multiples of the number of hosts were actually used in the experiment.

The experiment was carried out by using the RESQ2 queuing network analysis package [Sau82]. We show the response time versus the total number of terminals in Figure 7, and the total system (transaction) throughput versus the number of terminals in Figure 8.

5.2. Analysis of the Results

With our approximate models and under all of the assumptions we made, the distributed file system is better than the file server-based system when the system is less congested, i.e., when the number of terminals is low. The converse is true when the system is sufficiently congested. However, the differences between the two organizations in our experiment were not substantial, as shown in Figures 7 and 8. The cross-over point depends on the probability of remote disk access. The lower this probability, the higher the congestion a distributed file system can tolerate without losing in the comparison with a file server-based one. Both response time and throughput exhibit this phenomenon.

Results of this type are very useful in configuration design as well as in choosing a proper file system organization for a given workload. We can use these curves to select a minimum configuration that meets the throughput requirement and also the response time requirement by choosing a suitable number of hosts. For an existing distributed system, we can also easily check whether adding a few user terminals can increase the throughput enough to meet new demands while still keeping the response time below a reasonable limit.

In the particular system we experimented with, the bottlenecks were the host CPUs. Another experiment was carried out by hypothetically improving CPU speed by twenty percent in a file server-based system. The results of this proposed system are presented in Figures 9 and 10. It is interesting to observe that a 20% improvement of CPU speed may allow the system designer to solve the performance problem at hand simply by upgrading the CPUs without adding a new host. The tradeoff really depends on the relative costs of CPU upgrading and of the addition of a new host.

6. Conclusions and Future Research

A methodology has been presented for the initial design of a distributed system based on a local area network. The methodology can be used also to study alternative organizations and design tradeoffs. In particular, it has been applied to compare two file system organizations. The file server organization has the advantages of being conceptually simple and being able to achieve economies of scale. The distributed file system organization has the advantage of being robust against partial system failure. Our queuing network models attempt to compare the performances of these two different organizations quantitatively. As shown by our simple example, the tradeoff depends on the characteristics of the workload, i.e., on the locality of file accesses, and the utilizations of critical system resources.

Depending on the workload of a given system, a hybrid of these two file system organizations can be considered if there is a clear separation between heavily shared and lightly shared files. In this case, it may be better to put all the heavily shared files in a file server and distribute the rest of them over hosts.

We have not represented in our models the overhead due to the synchronization of accesses to shared files. The performance impact of serialization delays is being studied [Hać83]. It is intuitive to expect that the performance degradation will depend heavily on the granularity of locks, the amount of sharing, and the possible presence of bottlenecks elsewhere in the system. Models that reflect the replication of shared files for fast access are good candidates for future research in this area. The problem of consistency among replicated files needs to be examined with care.

We have used workload data collected in a single-machine system to model future distributed systems. Although the components of our distributed system models are validated and widely used in modeling computer systems, we will not be able to validate the distributed system model as a whole until systems of such type are constructed in practice. An earlier attempt of this sort can be found in [Gol83]. Even though the methodology we have outlined is typically performed off-line, we believe that it can be automated enough to become part of the distributed system so that periodic reallocation of shared files can be carried out to reflect major shifts in user behavior. Similarly, reallocation of user terminals or transaction processes can be employed to balance the load among various hosts in order to achieve better total system throughput and better response time.

Acknowledgement

We would like to thank Mark Hill for his kind suggestions on the preparation of illustrations.

References

[Bas75]
F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," Journal of ACM, 22, 2, April 1975, pp. 248-260.

[Cha75]
A. K. Chandra and C. K. Wong, "Worst-Case Analysis of a Placement Algorithm Related to Storage Allocation," SIAM J. Computing, 4, 3, September 1975, pp. 249-263.

[Cla78]
D. D. Clark, K. T. Pogran, and D. P. Reed, "An Introduction to Local Area Network," Proc. IEEE, 66, 11, November 1978, pp.1497-1517.

[Gol83]
A. Goldberg, G. Popek, and S. Lavenberg, "A Validated Distributed System Performance Model," Performance '83, edited by A. K. Agrawala and S. K. Tripathi, North-Holland, 1983, pp. 251-268.

[Hać83]
A. Hać, "On the Modeling of Queueing Networks with Serialization Delays," Computer Science Division Report, University of California at Berkeley, in preparation, 1983.

[Lee83]
T. P. Lee, "Configuration Design of Distributed Systems," Computer Science Division Report, University of California at Berkeley, in preparation, 1983.

[Mit82]
J. Mitchell and J. Dion, "A Comparison of Two Network-Based File Servers," Comm. of ACM, 25, 4, April 1982, pp. 233-245.

[Sal75]
H. M. Salkin, "Integer Programming," Addison-Wesley, Reading, Mass., 1975.

[Sau82]
C. H. Sauer, E. A. MacNair, and J. F. Kurose, "The Research Queueing Package, Version 2 : CMS Users Guide," IBM Research Report RA 139 (#41127), April 1982.

[Sch78]
A. L. Scherr, "Distributed Data Processing," IBM System Journal, 17, 4, 1978, pp. 324-343.

[Stu80]
H. Sturgis, J. Mitchell, and J. Israel, "Issues in the Design and Use of a Distributed File System," Operating Systems Review, 14, 3, July 1980, pp. 55-69.

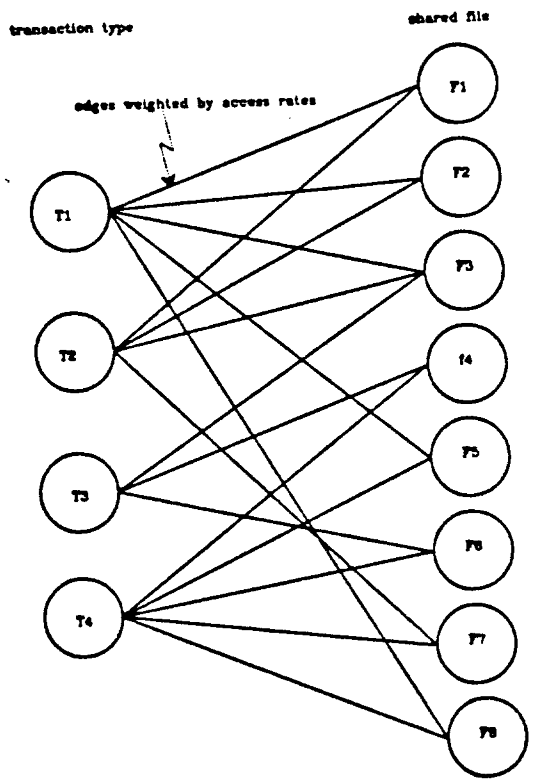

Figure 1. A Bipartite Graph Model of File Access

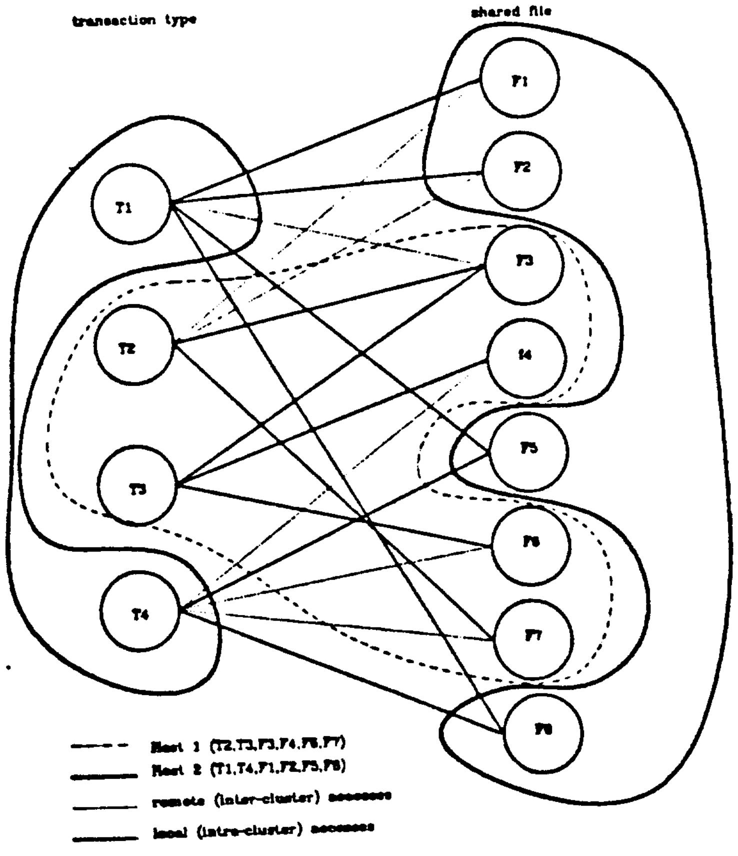
transaction type
shared file
T1
T2
T3
T4
F1
F2
F3
f4
F5
F6
F7
F8
Host 1 (T2,T3,F3,F4,F6,F7)
Host 2 (T1,T4,F1,F2,F5,F8)
remote (inter-cluster) accesses
local (intra-cluster) accesses

Figure 2. File Assignment

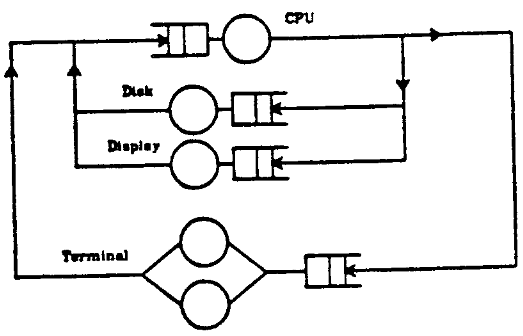

Figure 3. A Queuing Model for a Host

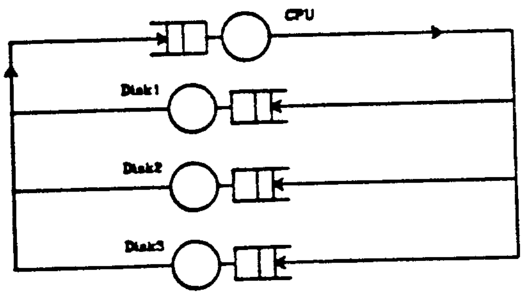

Figure 4. A Queuing Model for a File Server

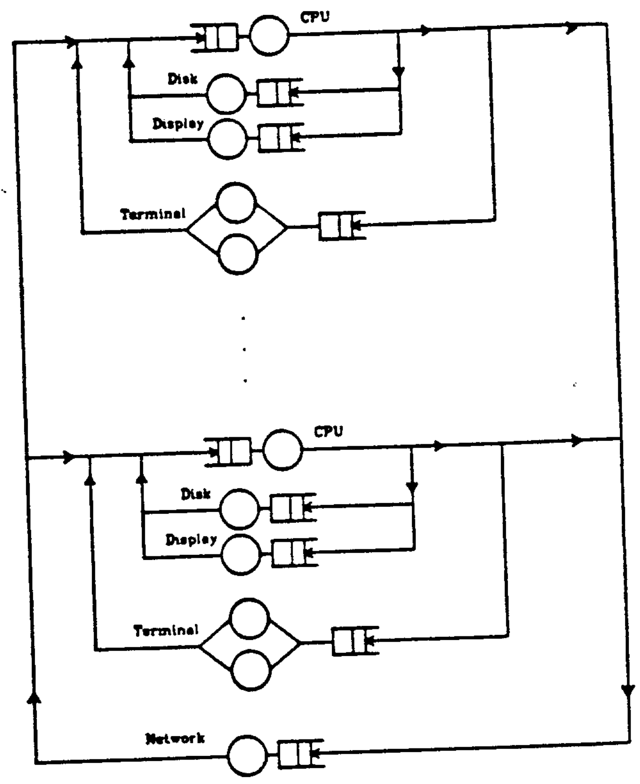

Figure 5. A Model of a Distributed File System

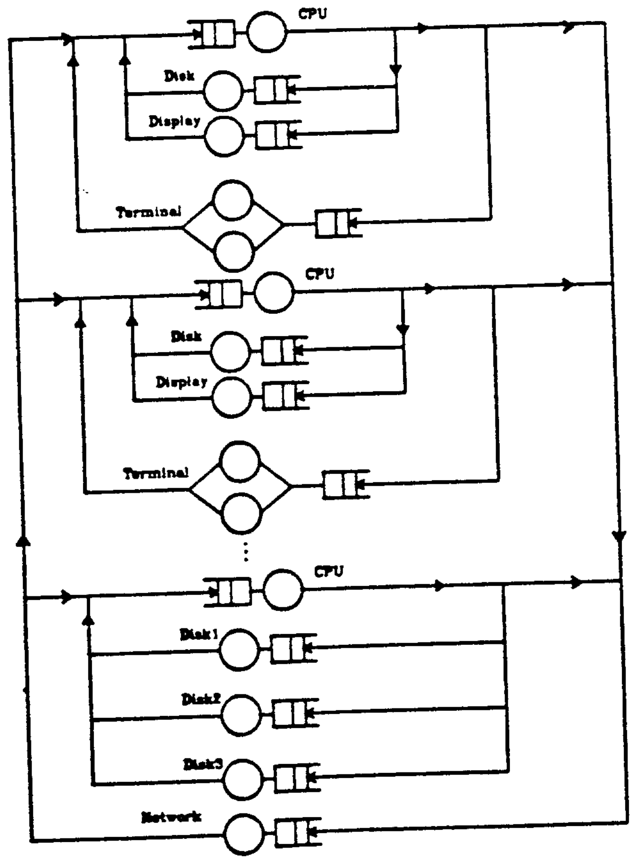

Figure 6. A Model of a File Server-based Distributed System

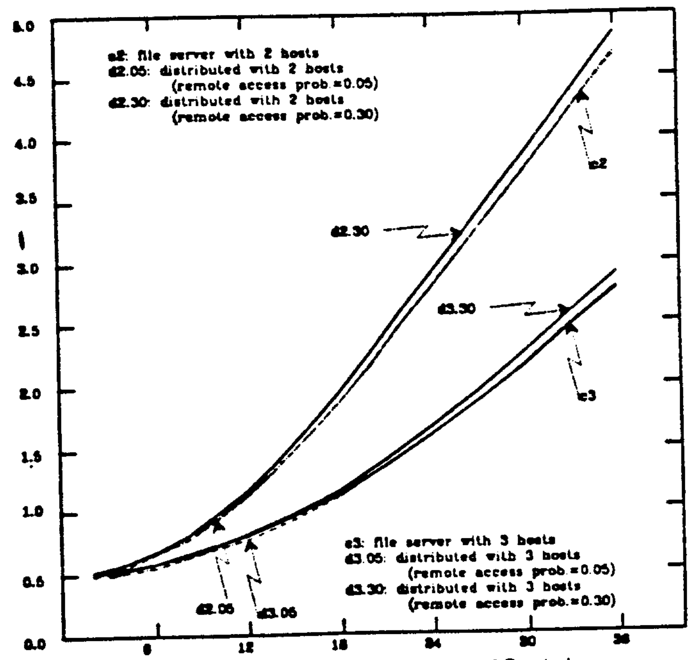

Figure 7. Response Time (sec) Versus Number of Terminals

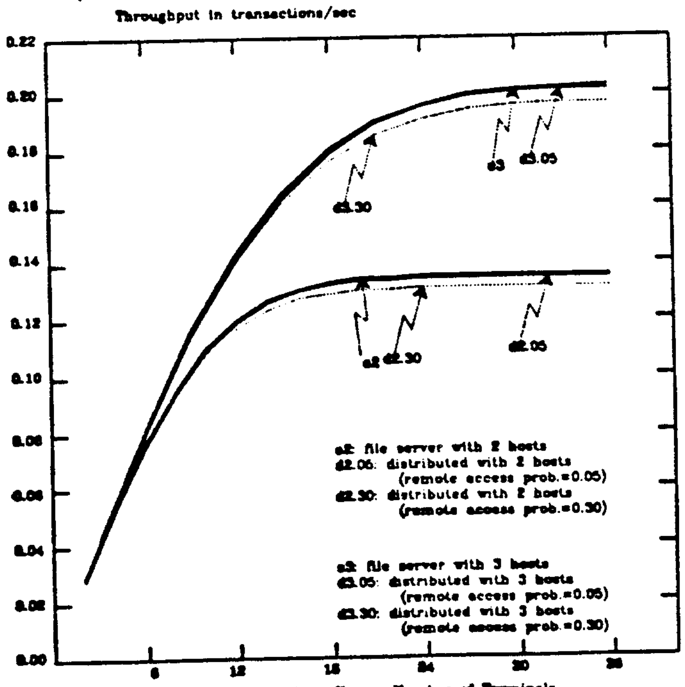

Figure 8. Throughput Versus Number of Terminals

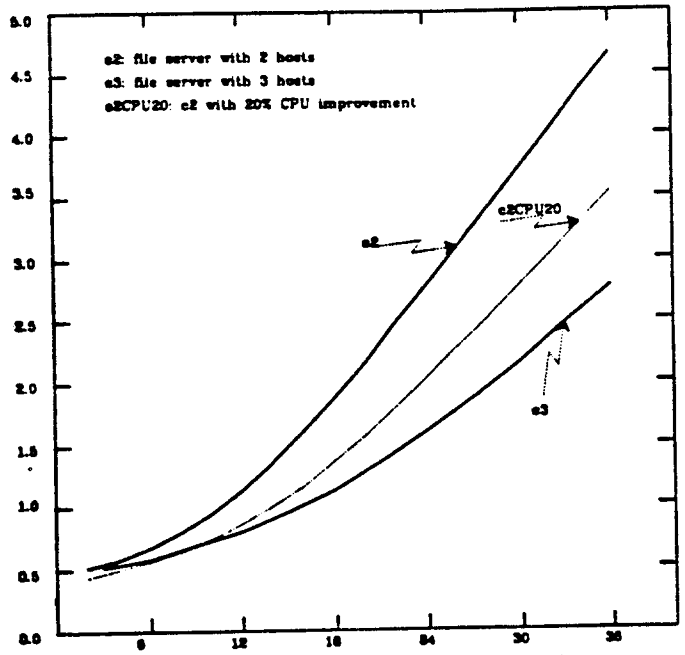

Figure 9. Response Time (sec) Versus Number of Terminals

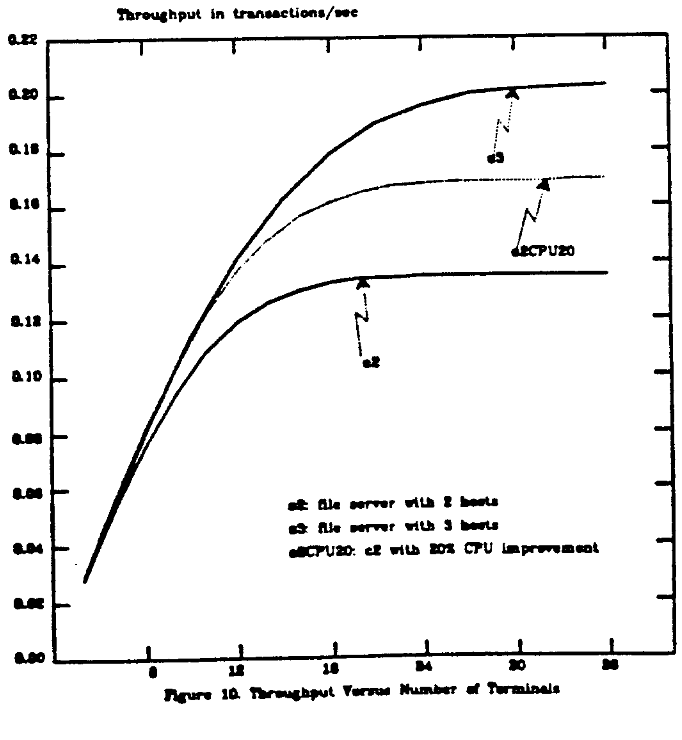

Figure 10. Throughput Versus Number of Terminals