A SIMPLE LAN FILE PLACEMENT STRATEGY

J. D. Padhye*, A. L. Rahatekar*, L. W. Dowdy*

In any multiuser distributed computing environment, there is usually a set of files which are accessed frequently by the user community and a set of disks connected via a local area network (LAN), used to store these files. It is the system administrator's job to decide which files are to be stored on each particular disk. Usually the disks available for storage are physically separated and connected via a network. A Network File System (NFS), handles all the file access requests. The disks typically have different characteristics since distributed networks usually have heterogeneous resources. The placement of heavily accessed files, across these disks, can have a large impact on the performance of the system in terms of the throughput. An effective file placement strategy should consider file access probabilities and disk characteristics. Placing all the files on the fastest available disk may not be optimal due to queuing and contention delays. Typical results of a poor file placement include low throughput and overloading of "hot" disks. Modeling a distributed system is not trivial, since it includes modeling the distributed computing resources, the network connecting them to the file servers, and the overhead introduced by the NFS. This paper presents a simple queuing network model of a distributed system and a method to efficiently calibrate the model with experiments. Certain approximations are made to simplify the analysis and calibration. Validation experiments comparing the predicted performance with the actual performance justify the approximations and indicate that simple models can serve the purpose of accurately indicating potential improvements in performance.

1 Introduction

In a distributed computing environment supporting multiple users, strategic placement of frequently accessed files can have a significant impact on the performance of the system. Disk drives at various locations across the network will typically have different characteristics and, thus, different service rates. In the absence of a proper file placement strategy, certain disks can become overloaded and overall system performance can be severely limited.

1.1 Approach Background

Considerable research has been conducted on the issue of file placement. Various models and surveys have appeared [DOWD82] [WAH84]. The approach examined in this paper is simple and straightforward. A client-server model is assumed [MENA94]. File requests are generated by a processor (i.e., a CPU) at one of the network nodes. Each request accesses the local area network and this network delay is modeled as a delay server with an experimentally observed mean delay time. After accessing the network, the request is serviced by the remote disk on which the requested file is stored. Each disk is modeled as a load-dependent server. Following service, the satisfied request returns to the CPU initiating the request. The goal is to place the files on the remote disks in a way that maximizes throughput across the LAN. The file placement directly affects the probabilities with which each remote disk is accessed.

This model is shown to accurately predict the performance of various file placement alternatives. The model helps in analytically determining the optimal file placement.

1.2 Motivational Example

Assume the following system parameters for the hypothetical system shown in Figure 1.

- Service rate of the CPU (μ_{cpu}) = 100 file requests/second.

*Department of Computer Science, Vanderbilt University, Tennessee, USA

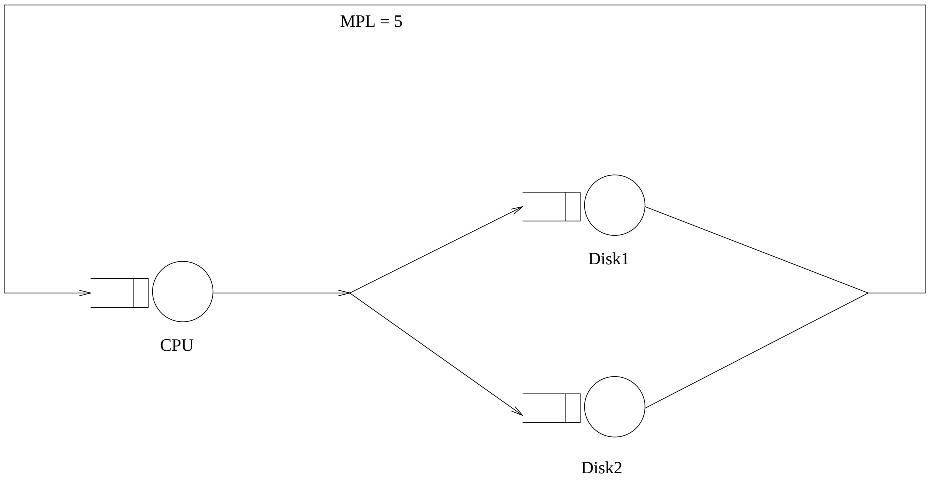

Figure 1: Example System

- Service rate of Disk 1 (μ_{disk1}) $=$ 10 file requests/second.
- Service rate of Disk 2 (μ_{disk2}) $=$ 5 file requests/second.
- Six files $File_1, \ldots, File_6$, are to be placed on these disks. The access probabilities for the files are 0.01, 0.09, 0.25, 0.25, 0.25 and 0.15, respectively.
- Multiprogramming Level (MPL) $= 5$. This represents the average number of file requests circulating in the example system.

The throughput of this system depends on the access probabilities for the two disks. Consider the case when $File_3$ and $File_4$ are placed on Disk 1, while the rest of the files are placed on Disk 2. In this configuration, each disk is accessed with 50% probability. For this file placement, the system throughput, calculated using MVA [REIS80], is 9.82 file requests serviced per second. However, if $File_3, File_4$ and $File_5$, are placed on Disk 1 (i.e., 75% of the file requests access the faster disk) and $File_1, File_2$ and $File_6$ are placed on Disk 2, (i.e., 25% of the file requests access the slower disk), the system throughput increases to 12.63 file requests serviced per second (i.e., a 29% increase in system throughput).

Overloading the fastest disk is detrimental to performance. For example, if all the files except $File_1$ are placed on Disk 1, (i.e., 99% of all file accesses are to the faster disk), the system throughput reduces to 10.10. By adjusting the access probabilities to the disks, the maximum throughput for this system is 12.69. This is achieved if Disk 1 is accessed 72% of the time and Disk 2 is accessed 28% of the time. However, with the given file access probabilities, and assuming that files are not split across the two disks, the optimal feasible file placement is to place $File_1$ and $File_3$ on Disk 2 and the remaining files on Disk 1. In this case, 74% of the load is to Disk 1 and 26% is to Disk 2. The resulting system throughput in this case is 12.67 file requests serviced per second.

This example illustrates the basic file placement approach. First, an appropriate model of the LAN system is constructed. Second, the optimal disk access probabilities are analytically determined. Third, a file placement is found which closely approximates the optimal disk access probabilities. As demonstrated, a significant performance impact can result from redistributing the files. Such redistribution is a relatively easy task.

The outline of the remaining paper is as follows. Section 2 describes the underlying queuing network model. Section 3 gives the basic file placement technique. Section 4 presents the parameters for an experimental system, used for validation purposes. Section 5 discusses the calibration of the model on the experimental system. Section 6 presents the experimental results. Section 7 compares the actual performance to the model predictions for various file placements.

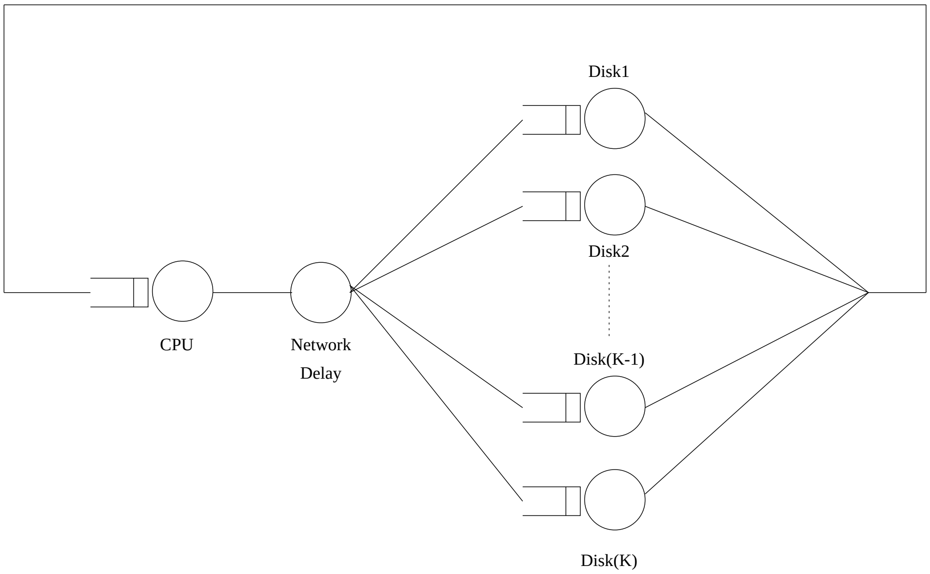

Figure 2: System Model

2 The System Model

The basic system model is shown in Figure 2. The system is modeled as a single class closed queuing network. It consists of a server modeling the local CPU, K remote disk servers, and a delay server which models the impact of the network. The network delay server models the fact that all file access requests are handled by the Network File System (NFS). Hence, each request is "slowed down" due to: 1) overhead of the NFS call processing, 2) network contention delay, and 3) network transmission delay. The service times associated with all servers are assumed to be negative exponentially distributed. A closed system, where the number of circulating customers (i.e., the number of file requests) remains constant is assumed and is represented by the multiprogramming level, MPL. The model parameters are summarized as follows:

1. MPL – multiprogramming level, the number of file requests in the system.
2. K – number of remote disks which can be used for file placement.
3. μ_{cpu} – service rate of the CPU. If all MPL file requests are not at the remote disks, then this represents the arrival rate of file requests to the remote disks.
4. μ_{disk_i} – service rate of Disk i.[1]
5. SP_i – array of size MPL for Disk i. Element j of this array represents the speedup of Disk i when there are j customers at the disk, relative to $j = 1$. This factor captures the load dependent nature of the disk servers. When there are multiple requests queued up at the disk, the seek time to the nearest request varies inversely with the number of queued requests. To illustrate, consider a single request for disk access. On average, the disk head must travel across one-third of the number of disk cylinders. If two requests are in the disk queue, the average number of cylinders to traverse to seek to the nearest requested cylinder reduces to one-fourth of the total number of cylinders. This progression continues as the number of queued requests increases. Thus, the effective service rate of a disk (i.e., the rate at which it can satisfy requests) changes as the number of queued requests changes. This implies that the disk is a load dependent server.
6. N – number of files to be placed across the K disks.

[1]The service rate is assumed to be the rate at which 500 bytes can be written to a disk, which is the average size of a file request.

7. P_i – access probability of Disk i
8. F_j – access probability of File j.

The access probability of a disk is dictated by the files resident on the disk. The disk access probability is calculated as the sum of the access probabilities of the files placed on that disk.

$$P_i = \Sigma F_j \quad \forall j, \text{ File } j \text{ is placed on disk } i \tag{1}$$

Without loss of generality, it is assumed that the disks are ordered by their service rates, with Disk 1 being the fastest.

$$\mu_i \geq \mu_{i+1} \quad \forall i, 1 \leq i \leq K \tag{2}$$

The CPU server is assumed to have a processor sharing queuing discipline. The disk servers are load dependent and are assumed to model a shortest seek time first (SSTF) queuing discipline. The method of calculating the load-dependent speedup for the disk servers is described in Section 5.

In a distributed system, processes make requests for file accesses via the processor on which they are executing. The processor forwards each request to the file server. Depending on the overall system workload, in highly I/O intensive networks, disk access time may be the bottleneck. In such systems, a reasonable approximation is to replace the distributed CPUs by a single central server [BUZE71]. This models a simple client-server relationship where the central server generates file requests for the remote disk servers. As seen in the experimental validation in Section 6, this simplifying assumption seems reasonable.

The system is considered to be a closed system with a fixed number (i.e., the multiprogramming level, MPL) of circulating file requests. This implies that the maximum number of outstanding file requests at the remote disks is MPL. The service rate of the central server determines what percentage of these MPL requests are at the remote disks and what percentage are in the process of being generated by the CPUs. Thus, the service rate of the central server determines whether the system is CPU-bound or I/O-bound.

3 The File Placement Algorithm

The algorithm to calculate the optimal file placement for a given system proceeds in two stages.

- Once the system queuing network model is constructed, the optimal branching probability to each of the disks is calculated based on a refined technique that uses load dependent MVA [REIS80].
- The algorithm assigns individual files to individual disks to closely approximate the theoretically optimal branching probabilities.

The details of the algorithm follow.

1. The optimal branching probability to each of the K disks is calculated. Load dependent MVA is used to calculate the throughput of the system for each of the possible ordered K-tuples $< P_1, P_2, \cdots, P_K >$, where P_i is the branching probability to Disk i. The output of this step is an ordered K–tuple, $< OP_1, OP_2, \cdots, OP_K >$, representing the optimal branching probabilities to each disk which maximizes throughput.

   It is noted that the search for the optimal branching probabilities is carried out over a potentially large search space, since each of P_i's can take values over the range $[0..1]$ while satisfying the constraint $\sum_{i=1}^{K} P_i = 1$. Several search refinements are possible. For instance, values for each P_i can be considered in steps of, say, 0.1, in the range $(0, 1)$. The solution can be continuously refined to any desired numerical precision. Hill climbing and other general search techniques are also applicable, since throughput as a function of the branching probabilities is continuous and well-behaved. Also, the fact can be used that the disk speeds are ordered implies that the optimal branching probabilities are likewise ordered. Additionally, since there are only a relatively small number of feasible file assignments, only those possibilities need to be analyzed by the model.

2. Once these theoretically optimal branching probabilities are known, a modified version of the *Bin Packing Using Best Fit* algorithm is

used to assign each file to a particular disk. The goal is to find a file assignment which closely matches the optimal branching probabilities. This algorithm works as follows:

(a) Consider K bins, each of which represents a disk. Bin i has a capacity equal to P_i.

(b) N items, each representing a file, are to be packed into these bins. Size of item j is equal to F_j.

(c) Each item j is placed in bin i such that

$$(P_i - F_j) = \min(P_k - F_j) \quad \forall k \; P_k \geq F_j \qquad (3)$$

(d) After the item j is placed in bin i, the capacity of bin i is modified

$$P_i = P_i - F_j \qquad (4)$$

(e) If no such i can be found then $i = 1$ i.e. the surplus files are assigned to the fastest disk.

4 A Specific Problem Instance

The system used for experimental validation, is a distributed system with four disks, connected via an ethernet and accessible through the NFS. A single file server services the requests for files located on all four disks. A queuing network model of the system is depicted in Figure 3.

4.1 Problem Data

The number of active processes is assumed to be constant at 15. The number of files to be placed across the four disks is taken to be 20 with access probabilities as shown in Table 2.

For this example:

- MPL = 15, the multiprogramming level.
- $N = 20$, the number of files.
- $K = 4$, i.e. four disks are under consideration, D_1, D_2, D_3 and D_4. Their service rates with a queue length of 1, are found out by experimental timing studies and are as shown in Table 1. An important consideration before running the experiments is the fact that the disks are cached. This implies that if the file access requests are implemented as reads, the first such request for a file will place it in the cache. Succeeding requests for that file will not cause any disk contention but will access the file from the cache. To eliminate the caching effects, each file access request is a write request. This modification ensures that all logical accesses correspond to physical accesses. Caching will further improve the performance of the system and the file placements suggested by this study would continue to provide performance improvements.

4.2 Model Solution

The technique described in section 3 calculates the optimal disk access probabilities [2] which result in the maximal throughput. These probabilities are shown in Table 3. The calculation of these optimal values is not accurate due to the unavailability of technical data for one of the disks. The file placement algorithm is used to approximate the optimal branching probabilities. The results are shown in Table 4. The access probabilities for the disks due to this file placement are an approximation to the calculated optimal values. This implies that the optimal throughput has to be recalculated. The theoretical model predicts that the throughput for the optimal file placement is 47.38. Along with this placement, nine alternative file placements are considered.

1. All files are placed on the fastest disk.
2. 90% of the file accesses are to the fastest disk with 10% to the next fastest. This is less extreme than configuration 1 but still relies heavily on the fastest disk.
3. All files are on the slowest disk. This helps evaluate the case where no disk characteristics or file placement strategies are considered.
4. The files are placed so that the access probability for each disk is equal to its relative base service rate. Thus, the load is spread across the

[2] The disk access probabilities are the branching probabilities $P_1, P_2, \cdots, P_K$ in Figure 2

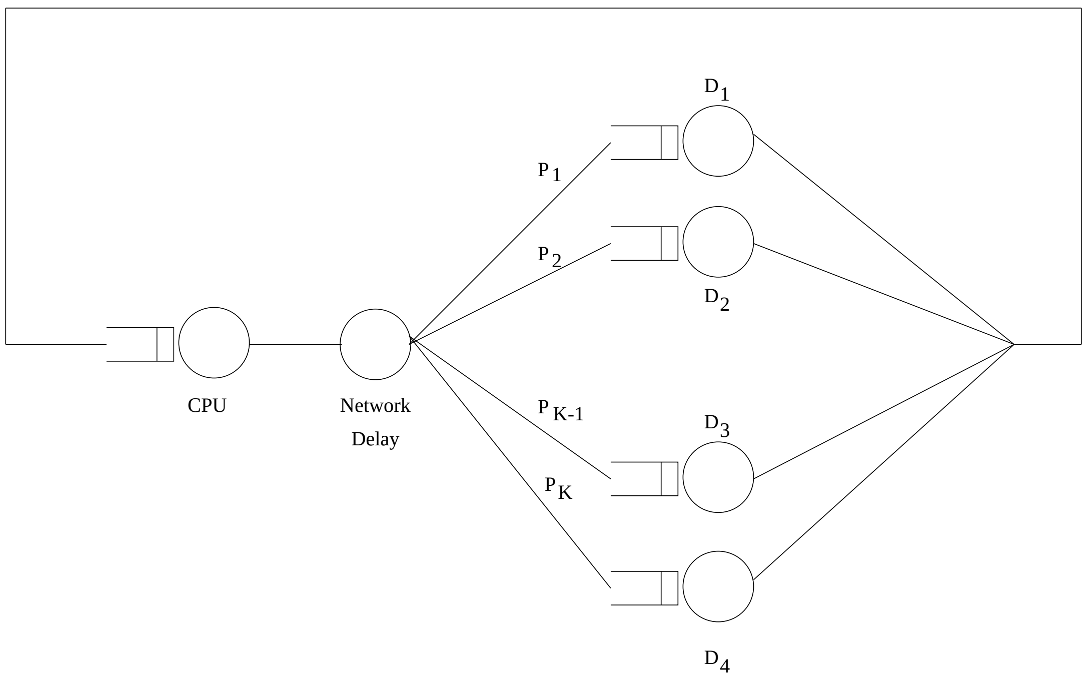

Figure 3: Experimental System

Disk	*Speed*
D_1	12.0250
D_2	10.8354
D_3	9.6339
D_4	9.1546

Table 1: Disk Service Rates, Queue Length = 1

File	*Access Probability*
FILE0.dat	0.0151
FILE1.dat	0.0943
FILE2.dat	0.0376
FILE3.dat	0.0296
FILE4.dat	0.0564
FILE5.dat	0.0074
FILE6.dat	0.0674
FILE7.dat	0.0922
FILE8.dat	0.0677
FILE9.dat	0.1026
FILE10.dat	0.0516
FILE11.dat	0.0510
FILE12.dat	0.0238
FILE13.dat	0.0000
FILE14.dat	0.0046
FILE15.dat	0.0270
FILE16.dat	0.0839
FILE17.dat	0.0284
FILE18.dat	0.0540
FILE19.dat	0.1042

Table 2: File Access Probabilities

Disk	*Access Probability*
D_1	0.2850
D_2	0.2525
D_3	0.2395
D_4	0.2230

Table 3: Optimal Disk Access Probabilities

Disk	*Calculated Access Probability*	*File Placement*	*Approximated Access Probability*
D_1	0.2850	4 18 11 2 17 15 12 5 14	0.2907
D_2	0.2525	16 8 6 3	0.2488
D_3	0.2395	1 7 10	0.2383
D_4	0.2230	19 9 0 13	0.2222

Table 4: The Optimal Configuration. Throughput = 47.38

disks in direct proportion to their service rates. This is a good load balancing heuristic.

5. The file accesses are evenly distributed between the two fastest disks.
6. The file accesses are evenly distributed between the two slowest disks.
7. The file accesses are evenly spread across the disks.
8. The file accesses are evenly distributed between the second and third fastest disks.
9. The file accesses are evenly distributed between the first and third fastest disks.

These file placements and their corresponding theoretical throughputs, for a file size of 500 bytes, are tabulated in Table 5. These results show the impact of selecting a good file placement.

5 Calibration of the Servers

The model is parameterized by evaluating the experimental system. This evaluation can be done in either of two ways:

1. A reasonable approximation of some of the parameters may be obtained by studying the underlying hardware. (e.g. an approximation of the speedup factors for the disks may be obtained on the basis of hardware parameters supplied by the manufacturer)
2. A synthetic workload can be used to measure the parameters. The workload is constructed and run on the system under consideration. Parameters like CPU and disk service rates can be obtained by actual measurements.

For this study, a combined approach was used. The empirical method was used whenever it proved difficult to get detailed hardware information.

The system considered has 6 servers visited by jobs. Their parameters are obtained as follows:

- CPU parameters

  The CPU service rate is obtained by using the experimental workload, removing all the I/O statements and measuring the average rate at which the CPU serviced the modified workload.

- Delay Server parameters

  The delay server represents network delay(ND). The service demand at the delay server is measured by making a write request across the network and then making the same request locally, from the machine to which the disk is attached. The difference is taken to be the service time requirement at the delay server. It is also possible though more complicated, to estimate this value from the network bandwidth, average load and the network access protocol.

- Disk parameters

  Disk access time, DAT, is made up of the following components:

  1. Seek time (ST)

  $$ST = \frac{X * ST_1}{(queue\ length\ +\ 2)} \quad (5)$$

     X is the total number of cylinders on the disk and ST_1 is the track-to-track seek time. When there is only one request to be serviced by the disk, the head has to travel, on an average, across one third of the total number of tracks on the platter. When there are two requests, on an average, it will have to travel across a fourth, to service the nearest request and so the time to complete the next file access request will decrease. The progression continues with an increase in queue length, resulting in the load-dependent nature of the disks.
  2. Latency time (LT)
  3. Settling time (STL)
  4. Data transfer time (DTT)

The service time T to write a file, is the cumulative time for sending a request across the network and getting the request serviced by the disk, and is given by,

$$T = ND + ST + LT + STL + DTT \quad (6)$$

The mean service rate of the disk [3] with one queued request is measured and the speedup values are calculated as the queue length increases.

[3] The inverse of the average service time.

Number	Alternative File Placements	Throughput
1	All files on the fastest disk	18.82
2	90% accesses to the fastest disk	20.87
3	All files on the slowest disk	14.62
4	Load balancing	46.66
5	All files on the two fastest disks	31.09
6	All files on the two slowest disks	26.64
7	Equal split	45.79
8	All files on the 2^{nd} and 3^{rd} fastest disks	29.37
9	All files on the fastest and 3^{rd} fastest disks	30.48

Table 5: File placements and associated throughputs

6 Experimental Results

The measured system throughputs for files of size 500, 600, 700 and 1000 bytes, for the optimal file placement and the nine alternatives tabulated in Table 5, are tabulated in Table 6. As a representative example, the theoretical throughput and the actual throughput for a file size of 500 bytes, are plotted for each file placement alternative to obtain Figure 4. The graphs for the other file sizes are similar in nature.

All readings have been obtained when the load on the network was relatively light. This enables maximum sensitivity of the throughput to the file placement since there is no other dominating factor.

7 Discussion of the Results

The actual and theoretical values of throughputs for the various file placements follow the same trend. From Figure 4 it is evident that the theoretically optimal configuration is also best according to the experimental results. Although there is some difference between the actual throughput and the values predicted by the model, the important point to note is that within the different file placements, the order of increasing throughput is the same for the theoretical and the actual results. This means that the model can predict the best configuration. Exceptions to this are the results of configurations 6 and 8. Theoretically configuration 6 will have less throughput than configuration 8 but the experiments claim otherwise. A similar discrepancy is also discernible between configurations 1 and 2. However these discrepancies are minor. The theoretical throughputs of three configurations out of the ten considered are all close to the optimal, and the actual results are in agreement with this. These three are:

1. Configuration 1, the optimal file placement according to the model;
2. Configuration 4, load balancing;
3. Configuration 7, all the four disks are uniformly accessed. This configuration works well but intuitively will not work in case the disk speeds are widely disparate unlike the system on which the experiments have been run.

The near-optimality of configuration 4 suggests that maybe the brute force calculation for the optimal branching probabilities can be replaced with an effective heuristic.

The results indicate that the simple model accurately predicts the optimal file placement. The performance gains over the worst possible assignment are seen to be quite significant, of the order of 225%, configuration 3 has a throughput of 12.95 jobs/sec while the optimal configuration gives a throughput of 42.31 jobs/sec.

7.1 Conclusion

File placement can improve the throughput and hence response time significantly [4]. The experiments indi-

[4] The assumption of a closed system implies fixed MPL and so throughput and response time are inversely proportional according to Little's Law

Configuration	*File Size = 500*	*File Size = 600*	*File Size = 700*	*File Size = 1000*
Optimal	42.31	42.91	41.69	36.90
1	18.95	17.93	17.81	17.50
2	18.74	19.93	19.90	19.54
3	12.96	13.27	13.14	13.05
4	42.36	43.04	41.39	37.46
5	23.75	25.36	24.55	24.27
6	26.51	25.54	26.18	25.95
7	42.33	42.74	41.06	37.38
8	23.40	24.87	23.89	24.41
9	27.87	28.86	29.36	28.24

Table 6: Experimental Results

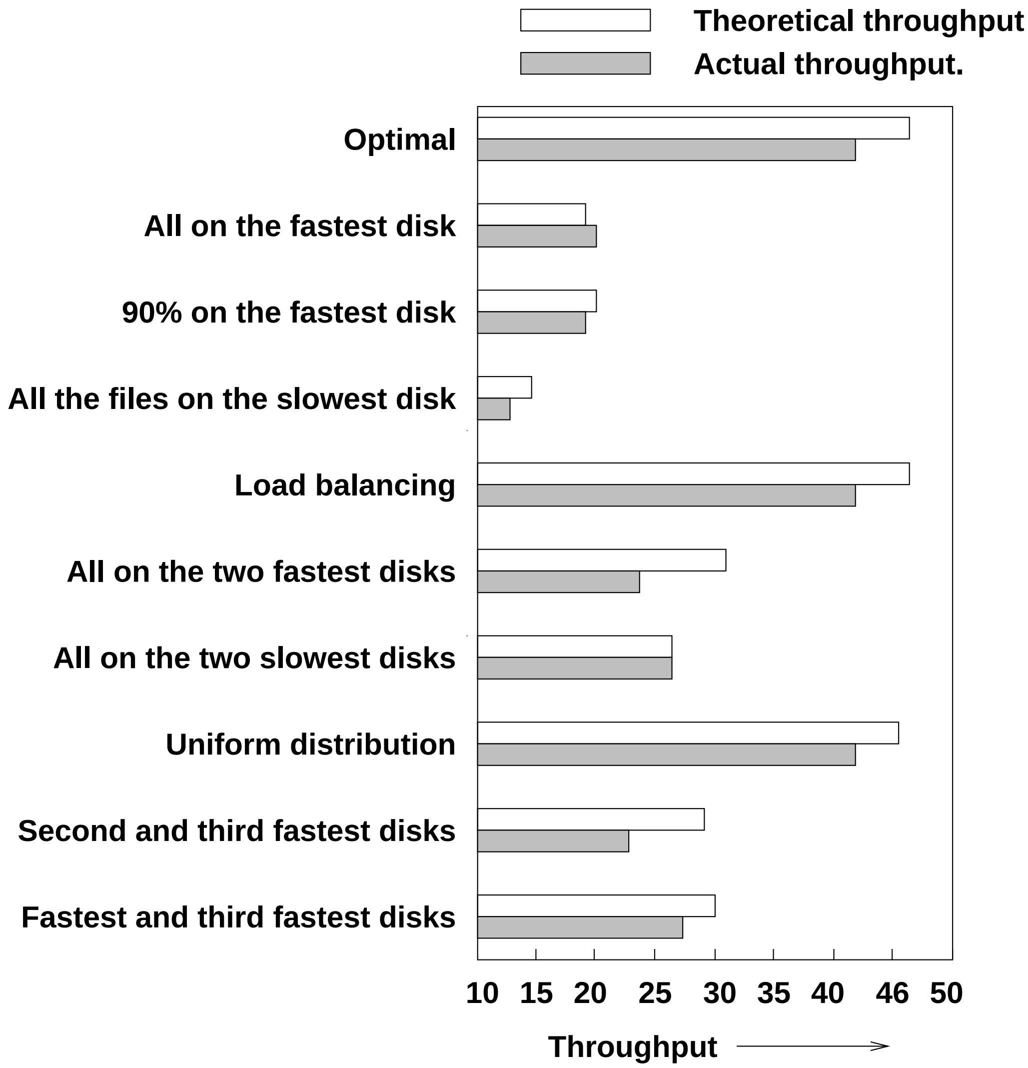

Figure 4: Comparison of model predictions and experimental results, file size = 500 bytes

cate that is not necessary to have a complicated model of the distributed system, with extensive calibration, to realize the potential performance gain. Matching trends between the theoretical and experimental results suggests that the calibration of the disks was accurate. The process of file placement can be simplified by eliminating the search for the optimal disk access probabilities and using the load balancing heuristic. Since the network delay is assumed constant, the removal of the network delay server will only increase all throughputs by a constant. If the only objective is find the optimal file placement accurately and not the actual throughput, the network delay server is not necessary. Future research into the optimal file placement of a set of files, on a particular disk, will lead to an integrated file placement solution for distributed systems.

8 Acknowledgments

The authors would like to thank Douglas Voss and P. Krishnan for their help towards improving the authors' understanding of the local network.

References

[BUZE71] J.P. Buzen, "Queuing network models of multiprogramming", *Ph.D Thesis*, Div. of Eng. and Appl. Sci., Harvard U., Cambridge, Mass., 1971.

[DOWD82] L.W. Dowdy and D.V.Foster, "Comparitive models of the file assignment problem", *ACM Computing Surveys*, 14, 2 (June 1982), pp. 287–313.

[MENA94] D.A. Menascé, V.A.F. Almeida, and L.W. Dowdy, **Capacity Planning: A Practical Approach – From Mainframes to Client-Server Systems**, Prentice-Hall, 1994.

[REIS80] M. Reiser and S.S. Lavenberg, "Mean value analysis of closed multichain queuing networks", *J.ACM*, 27, 2 (April 1980), pp. 313–323.

[WAH84] B.W. Wah, "File placement on distributed computer systems", *IEEE Computer*, (January 1984), pp. 23–32.