Proceedings of Sixth ACM Symposium on Operating Systems Principles (November 1977) 85-90.

MEASUREMENTS OF SHARING IN MULTICS

Warren A. Montgomery

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts 02139

ABSTRACT

There are many good arguments for implementing information systems as distributed systems. These arguments depend on the extent to which interactions between machines in the distributed implementation can be minimized. Sharing among users of a computer utility is a type of interaction that may be difficult to provide in a distributed system.

This paper defines a number of parameters that can be used to characterize such sharing. This paper reports measurements that were made on the M.I.T. Multics system in order to obtain estimates of the values of these parameters for that system. These estimates are upper bounds on the amount of sharing and show that although Multics was designed to provide active sharing among its users, very little sharing actually takes place. Most of the sharing that does take place is sharing of system programs, such as the compilers and editors.

CR Categories: 4.35, 4.6

Keywords and Phrases: Distributed Systems, Multics, Program Behavior, Virtual Memory, Measurements, Sharing.

1. Motivation

One of the driving forces in the development of large, central computer systems was the high cost of computer hardware. Current trends in electronic technology indicate, however, that hardware costs are no longer the dominant factor in the design of an information system.

There are many reasons to implement a large information system as a distributed system, with several independent, communicating machines. These reasons include:

I. Increased reliability. Each machine in a distributed system would be smaller and simpler than the single machine in an equivalent central system. Thus the individual machines would be less likely to fail, and a failure in one machine need not disrupt all system operations.

II. Ability to grow. Adding a new machine to a distributed system can increase the computing power available without affecting the existing system.

III. Distributed authority. The availability and protection of information managed by each machine in a distributed system can be independently controlled.

IV. Better response. The parallelism available in a distributed system allows it to respond more rapidly than a multiplexed centralized system. Communication delays also can be reduced by placing the information and computing power physically near where they are needed.

All of the arguments for choosing a distributed organization for an information system are dependent on the extent to which that system can be separated into independent components. Many of the advantages of a distributed system disappear if a significant amount of essential information is shared by components that are implemented on separate machines.

Sharing of information by processes executing on separate machines requires communication in order to keep the shared information consistent. Some protocols for maintaining consistency have been developed recently, but the problem is still poorly understood [1], [2], [3].

Current centralized systems can be divided into components, such as the files and processes belonging to individual users of a computer utility. However, the extent to which the components are independent is uncertain, because information is shared among components. Thus, before considering a distributed implementation for an information system, it is important to know the

This research was performed in the Computer Systems Research Division of the M.I.T. Laboratory for Computer Science. It was sponsored in part by the Advanced Research Projects Agency (ARPA) of the Department of Defense under ARPA order No. 2095 which was monitored by the Office of Naval Research under contract No. N00014-75-C-0661.

nature of sharing in that system. This paper reports an attempt to measure sharing in a large, centralized computer system: specifically, the Multics system at M.I.T.

The Multics system is a good choice for such measurements because it was designed to provide extensive facilities for controlled sharing, more so than other commercially available computer systems. Thus an examination of Multics should provide an upper bound on the amount of sharing in current systems.

The data reported here were gathered on the M.I.T. Multics system. However, the measurement tools developed could easily be used on other Multics sites.

2. Definitions

Before attempting to measure sharing, it is necessary to define what should be measured. A shared unit is called an object, which refers to a logical unit of information. The definition of sharing that follows is an attempt to capture the aspects of sharing that may be difficult to provide in a distributed system.

> An object is said to be shared in a time window t if it is referenced by at least 2 different processes within the time t.

It is useful to distinguish two kinds of references to an object: writes, which modify the contents of an object, and reads, which do not. This distinction allows us to define two specialized kinds of sharing: write-sharing, in which an object is shared and is also written by at least one process, and read-sharing, in which all references to the object are reads. Write-sharing is inherently the more difficult of the two to provide in a distributed system, because it requires communication among the processes that are referencing the shared object, while read-sharing does not.

The time window in the definition of sharing is a very important parameter, since it is an indication of the communication bandwidth needed to provide sharing. A much higher bandwidth is needed to provide sharing in a small time window than is needed to provide sharing in a large time window. The shared memory of a centralized system provides a very high bandwidth, while the bandwidth available in a distributed system is generally lower.

Finally, it is necessary to measure the sharing of different types of objects separately, because different types of objects are referenced in different ways, some of which may be more difficult to provide in a distributed system. The types of objects considered here are segments, directories, and pages.

3. Goals

Having chosen a definition of sharing, we can now ask what parameters best characterize the amount of sharing that takes place in a computer system. The parameters described below appear to be most useful for this purpose.

One parameter that is definitely of interest is the percentage of all objects of each type supported by the system that are shared in a given time window. The magnitude of this parameter increases as the time window is increased. An infinitely long time window gives the percentage of all objects of a given type that are shared at some point during their lifetimes. This parameter can be used to decide whether or not provisions for sharing should be made at the time that an object of that type is created. Smaller time windows give percentages of objects involved in more active sharing.

A second parameter of interest is the percentage of all object references that reference shared objects. This parameter provides an indication of the number of references that may require special treatment in a distributed system.

Another parameter that we wish to measure is the average number of processes that share a typical object. If a shared object is implemented by a multiple-copy strategy, then this parameter can be used to estimate the number of copies required and the communication that may be needed to keep the copies consistent.

4. Measurement Techniques

Knowledge of the parameters described above would allow us to estimate the storage capacity and communication bandwidth needed to provide sharing in a distributed system. Unfortunately, the Multics system does not provide the information necessary to measure these parameters directly: they must be indirectly measured or estimated. This section outlines the feasible measurements, and how these measurements relate to the parameters described above. It also provides a brief description of the virtual memory management algorithms used in the Multics system, which yield some data on the use of objects by processes. More detailed descriptions of these algorithms and the databases that they maintain appear in [4] and [5].

The measurements reported were made by examining supervisor data bases that describe the use of objects by processes. These databases were copied from a running system to produce a self-consistent snapshot of the state of the system at one instant in time. Forty such snapshots were analyzed to obtain the data presented here.

The Multics supervisor provides three primitive types of objects: segments, which are logical information containers holding up to 261120 36-bit words, directories, which form the hierarchy used to organize segments, and pages, which are information containers holding 1024 36-bit words. Each page is part of some segment or directory and contains part of the information content of that segment or directory. The Multics virtual memory mechanism maintains detailed information about the use of segments and directories by processes for approximately 1000 segments and directories. This information is contained in a software-implemented cache, known as the Active Segment Table (AST), which is managed by an LRU algorithm [4]. The segments and directories that are described in this cache are referred to as active. For the system being measured, the average length of a active segment was 8.6 pages, while the average length of an active directory was 3.9 pages.

For each active segment, the AST contains a list of the processes that have referenced that segment since it last became active. If two different processes appear on the list for a particular segment, then that segment has been shared, by the definition of sharing given above, in the length of time that the segment has been active. Unfortunately, the length of time that a particular segment has been active cannot be

determined directly from the available information and must be estimated. This time varies from a minimum of about 1 minute to several hours. The mean time is about 6 minutes.

Similar information is available for the active directories. Processes do not frequently reference directories explicitly, but a reference to a segment implicitly references all directories above that segment in the file system hierarchy. Because of these implicit references, and the LRU management algorithm, directories tend to remain active longer than do segments. The average length of time that a directory was active in the system being measured was about 25 minutes.

The amount of write-sharing of segments can be estimated by examining the access privileges that processes have to shared segments. (This only gives an upper bound on the amount of write-sharing, as a process that has write access to a segment does not necessarily write that segment.)

There is less information available about the use of pages by processes. An estimate of the sharing of pages can be made by considering a page to be shared if the segment or directory of which that page is part is shared, that is, all pages of shared segments and directories are considered to be shared. This estimate is again an upper bound, as most segments and directories have more than one page, and all processes that reference a segment or directory do not reference all of its pages.

We can easily estimate the magnitude of two of the sharing parameters described in the previous section from the data described above: the percentage of objects that are shared, and the average number of processes that share an object. A good estimate of the percentage of object references that reference shared objects cannot, however, be obtained from the available data.

5. Results

This section reports estimates of some of the sharing parameters described in section 3. The results reported in this section are averages obtained from 40 snapshots of the M.I.T. Multics system taken during the spring and summer of 1977. Except as noted below, estimates based on data from any single snapshot were within 10% of the reported averages.

5.1. The Sharing of Segments and Directories

Table 1 shows the total numbers and percentages of the active segments and directories that were shared in all 40 snapshots. The table shows that about 57% of the active directories were shared, while only 12% of the segments were shared.

Table 1
Sharing of Active Segments and Directories.

	Unshared	Shared
Segments	21037 (88%)	2910 (12%)
Directories	4467 (43%)	5980 (57%)

One reason for the larger proportion of shared directories is that the implicit references to directories cause a directory to be shared if two processes are using segments below that directory. Thus directories near the top of the file system hierarchy tend to be shared.

System processes, which perform such functions as making backup copies of recently modified segments and creating processes for users, reference many directories but do so very infrequently. These references contribute to the high level of measured directory sharing, but would probably not contribute to sharing across machine boundaries in a distributed system. The references that cross system boundaries could be minimized by using different processes to perform system functions at each site. A later section of this report discusses the effects of such system processes on our measurements.

5.2. Read-Sharing and Write-Sharing of Segments

Table 2 gives the numbers and percentages of active segments that were read-shared or write-shared.

Table 2
Read-Sharing and Write-Sharing of Segments

Read-Shared	2505	(10.5%)
Write-Shared	405	(1.7%)

These figures suggest that very little write-sharing of segments takes place. An examination of the write-shared segments showed that most were being used to implement sequential communication between processes, (such as mail, or resource requests), which could easily be provided in a distributed system without shared memory.

5.3. The Effect of System Processes and Segments

The measurements of sharing presented above include all of the processes, segments, and directories in the Multics system. We would like to be able to examine sharing of objects by user controlled processes separately from sharing due to the system processes mentioned above.

It is also interesting to examine the sharing of user-created objects separately from that of system objects such as system programs or subroutine libraries. Many system objects are very rarely modified, and therefore could easily be duplicated at each site in a distributed system; or they are directly related to the functioning of the machine, and thus would not be shared across site boundaries in a distributed system.

Table 3 gives the results of performing some of the measurements reported in Tables 1 and 2 on the same system (a) with the effect of system processes removed, and (b) with both the system processes and system segments and directories removed. The table gives the total numbers and percentages of read-shared and write-shared active segments and of active directories in the 40 snapshots. Three sets of figures are given, one for each of the two restricted cases mentioned above, and a summary of the results reported in Tables 1 and 2 for comparison.

Table 3
The Effect of System Processes and System Objects on Sharing Measurements

	All Processes, All Objects	User Processes, All Objects	User Processes, User Objects
Read-Shared Segments	2505 (10.5%)	2236 (10.6%)	308 (2.1%)
Write-Shared Segments	405 (1.7%)	180 (0.8%)	24 (0.2%)
Shared Directories	5980 (57%)	2899 (31%)	2476 (24%)

These figures show that the system processes are responsible for a substantial proportion of the sharing of directories, and most of the write sharing of segments. They also show that most of the read-shared segments are system segments. Each process references a large number of system programs. The segments that contain these programs account for most of the read-sharing.

5.4. The Average Number of Processes that Share an Object

The figures presented so far suggest that sharing of segments may not be very important in Multics. The percentages of shared segments seen here suggest that relatively rarely is a particular segment shared. The third of the sharing parameters discussed in section 3, the average number of processes that share an object, presents a different picture.

Table 4 shows this average for the three cases indicated in Table 3. The figures in parentheses show the average for the shared objects only, while the first set of figures shows the average for all objects, shared or unshared.

Table 4
The Average Number of Processes Sharing an Object

	Segments	Directories
All Processes, All Objects	2.4 (12.5)	4.1 (6.5)
User Processes, All Objects	2.4 (13.2)	3.4 (8.7)
User Processes, User Objects	1.1 (3.4)	2.2 (3.1)

The figures in Table 4 show that although few segments and directories are shared, those that are shared are widely shared. This seems to be true for user-created objects as well as for system-created objects, although the averages for the user-created objects are lower.

Figure 1 is a histogram of the data used to derive the average for all processes sharing all segments. It shows the number of segments that were shared by exactly N processes as N varies from 1 to 50. Notice that most of the segments are unshared or shared by only a few processes, while a few segments are shared by most of the processes on the system. These few are the segments that contain the most popular system programs. These widely-shared segments account for the large averages reported in Table 4.

Figure 1
The Number of Segments Shared
by Exactly N Processes

```
N
 1(21037) XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
 2  (952) XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
 3  (289) XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4  (221) XXXXXXXXXXXXXXXXXXXXX
 5  (134) XXXXXXXXXXXXX
 6   (98) XXXXXXXXXX
 7   (89) XXXXXXXXX
 8   (77) XXXXXXX
 9   (57) XXXXXX
10   (58) XXXXXX
11   (37) XXXX
12   (26) XXX
13   (33) XXX
14   (35) XXXX
15   (31) XXX
16   (25) XX
17   (23) XX
18   (21) XX
19   (16) XX
20   (16) XX
21   (11) X
22   (22) XX
23   (24) XX
24   (13) X
25   (11) X
26   (23) XX
27   (18) XX
28   (27) XXX
29   (17) XX
30   (21) XX
31   (22) XX
32   (25) XX
33   (20) XX
34   (22) XX
35   (33) XXX
36   (30) XXX
37   (25) XX
38   (22) XX
39   (41) XXXX
40   (43) XXXX
41   (47) XXXXX
42   (37) XXXX
43   (38) XXXX
44   (27) XXX
45   (17) XX
46   (12) X
47   (11) X
48    (5)
49    (7) X
50+  (21) XX
```

5.5. Sharing of Pages

As previously noted, information on the sharing of pages by processes is not directly available, and must be estimated based on the use of segments and directories. The three-level memory hierarchy used by Multics allows us to distinguish three sets of pages for which sharing parameters can be estimated. Each level of the memory hierarchy is managed by a page replacement algorithm that closely approximates LRU, so as to keep the most recently referenced pages in the lowest levels of the hierarchy. In the snapshots analyzed in this report, all pages in main memory (core) had been referenced within the last 0.5 second, and all pages in the second level of the hierarchy, the paging device (PD), had been referenced in the last 4 minutes. Pages in the third level (disk) had not been referenced in the last 4 minutes. (These time intervals, which indicate the time at which a page was last referenced by any process, should not be confused with the time window of sharing, which is the same for segments and directories as for their pages.)

Table 5 reports estimates of the amount of read-sharing and write-sharing of the pages in the fastest levels of the memory hierarchy.

Unlike most parameters, the amount of sharing of pages observed in the individual snapshots varied considerably from the averages reported here. The percentage of pages in main memory that were read-shared, for example, varied between 15% and 57%. The contents of memory at the time of a snapshot depend strongly on the references to pages that were made within 0.5 second of the time of the snapshot. Few processes run in that time interval, and the references made in that time interval may not be typical.

Table 5
Sharing of Pages

	Referenced in Last 0.5 Second	Referenced in Last 4 Minutes
Read-Shared	1569 (29%)	12025 (23%)
Write-Shared	67 (1.3%)	874 (0.8%)

Notice that the pages more recently referenced are more likely to be shared. We expect that shared pages are more frequently referenced, as they are referenced by more than one process.

We can also estimate the average number of processes sharing a page at each level of the memory hierarchy in the same way as was done for segments and directories in section 5.4. Table 6 presents these estimates.

Table 6
The Average Number of Processes Sharing a Page

	Pages of Segments	Pages of Directories
Core Pages	7.1	26.3
PD Pages	4.7	16.1
Disk Pages	2.1	5.4
All Pages	2.7	7.9

Notice that the number of processes that share a page is highest in the fastest portion of the memory hierarchy. This is because the chance that a page will be referenced increases with the number of processes that share that page, and thus shared pages compete more effectively for space on the paging device and in core.

It is interesting to note that the average number of processes sharing pages of segments is larger than the average number of processes sharing segments. This observation suggests that the probability that a particular page is shared does not depend on the size of the segment that contains that page. Thus the more pages that a segment contains, the more likely it is that that segment is shared. As all pages of a shared segment are deemed to be shared, such a tendency would account for the larger averages for pages than for segments.

6. Conclusions

The results in the previous section show that very little sharing of user-created objects takes place among user-controlled processes on the M.I.T. Multics system. This fact suggests that it may be possible to implement a Multics-like system in a distributed environment where sharing of memory pages is not possible.

There is very little write-sharing of segments. Most of the write-shared segments are referenced only by system programs, which use simple protocols to synchronize their writes.

Although a higher proportion of directories than of segments are shared, it is unlikely that shared directories would cause as much trouble in a distributed system as would shared segments.

Updates to directories are performed by the supervisor, which could be modified to provide the same functionality in a distributed system. Updates to segments, however, are made by user written programs, which rely on the fact that all processes share the same physical copy of a segment in order to synchronize their references. It would be difficult to provide the same functionality in a distributed system.

The figures presented suggest that although it would be possible to build a "distributed Multics", such a configuration may not be economically feasible. The segments and directories that are shared by all processes would need to be present at each site in such a system. Thus a distributed computer utility would need much more memory than a centralized utility to support the same number of users.

The measurements reported here are, as noted, upper bounds on the amount of sharing in a computer utility: the actual level of sharing may be much

lower. With better information about the actual patterns of references to objects made by individual processes, the working set model [6] could be used to give accurate estimates of the time window in which each object is being shared. Such information would allow us to measure accurately the sharing parameters described in Section 3. Accurate measurements would be very useful in determining the amount of extra memory that would be needed to support a distributed computer utility.

More accurate measurements would not, however, affect the conclusion that the level of sharing in a computer utility does not logically preclude a distributed implementation.

Acknowledgements

This study was suggested by Professor J. H. Saltzer, whose advice during the course of the work is also greatly appreciated. Several members of the Computer Systems Research division of the Laboratory for Computer Science made very useful suggestions as the metering program was being designed; I would particularly like to thank Alan Luniewski and Drew Mason. Preliminary drafts of this report were read by Professor Saltzer, Professor L. Svobodova, and the reviewers, whose comments have contributed to the more concise and effective presentation of the data and discussion of technical details. Finally, I would like to thank my wife, Carla, for her encouragement and for her help in making this report clearer and more readable.

References

1. Alsberg, P.A., Belford, G.G., Day, J.D., Enrique, G. Multi-copy resiliency techniques. University of Illinois, Center for Advanced Computation Document No. 202 (May 1976).

2. Johnson, P.R., and Thomas, R.H. The maintenance of duplicate databases. Arpanet Network Working Group RFC No. 677 (January 1975).

3. Thomas, R.H. A solution to the update problem for multiple copy data bases which uses distributed control. Bolt Beranek and Newman, Inc. Technical Report No. 3340 (July 1976).

4. Bensoussan, A., Clingen, C.T., and Daley, R.C. The Multics virtual memory: concepts and design. Comm. ACM 15, 4 (May 1972), 308-318.

5. Organick, E.I. The Multics System: An Examination of its Structure. MIT Press, Cambridge, Mass, 1972.

6. Denning, P.J., The working set model for program behavior. Comm. ACM 11, 5 (May 1968), 323-333.