An Analysis of Trace Data for Predictive File Caching in Mobile Computing

Geoffrey H. Kuenning, Gerald J. Popek, Peter L. Reiher
University of California, Los Angeles[*]

Abstract

One way to provide mobile computers with access to the resources of a network, even in the absence of communication, is to predict which information will be used during disconnection and cache the appropriate data while still connected. To determine the feasibility of this approach, traces of file-access activity for three diverse application domains were collected for periods of over two months. Analysis of these traces using traditional and new measures reveals that user working sets tend to be small compared to modern disk sizes, that users tend to reference the same files for several days or even weeks at a time, and that different users do not tend to write to the same file except in highly constrained circumstances. These factors encourage the conclusion that an automated caching system can be built for a wide variety of environments.

1 Motivation

The value of mobile computers is that they allow users to work while disconnected from their normal resources. However, mobile computers typically have a great deal less disk storage than is available via remote mounting on connected networks. This forces mobile computer users to face a challenging problem of making sure their limited disks always store the information they will need while disconnected from other machines. Requiring users to deal explicitly with this issue puts a heavy burden on them, and the realities of modern software methods make it nearly impossible for users to identify all the files they actually need.[1] A fully automated caching mechanism that predictively stored all files a user needs on his mobile machine would be very valuable. Such a mechanism is only practical, however, if information that can be gathered automatically fully captures the typical user's working set of files.

A prototype system of this sort was developed under CMU's Coda system [6, 14] and proved successful, but was inconvenient for the user and was tested only in one application environment.

We undertook this research to investigate the practicality of automatic file caching for mobility in a wider set of application domains, and to discover new and less-burdensome ways of identifying files to be cached. Our approach was to collect traces of file-access activity in several environments over a long period of time, and analyze them for feasibility and predictability of caching.

We chose to collect our own traces, rather than using existing traces, for three reasons. First, few existing traces are long enough. Because most existing traces collect read/write activity, a few weeks of data is sufficient to tax resource limits. We were interested in observing longer-term periodic behaviors such as end-of-the-month billing work in an accounting department, which therefore required a several-month trace to establish a pattern.

Second, existing traces have tended to be limited to an engineering application domain, usually programming. We wanted to investigate the behavior of non-programmers as well, in the twin beliefs that this type of user will eventually be the largest population of portable users, and that these users may behave quite differently from programmers.

Third, most previous studies have generally been limited to analysis of working-set sizes and file-system performance data [1, 2, 6, 11, 14]. The latter is not relevant to this research, and the former, while very important, is not in itself sufficient to characterize the user behaviors critical to successful mobile caching.

Successful automated caching requires two charac-

[*]This work was partially supported by the Advanced Research Projects Agency under contract N00174-91-C-0107.

[1]For example, starting the X Window System requires access to 10–30 files or more. The identities of many of these are surprising even to expert systems programmers [6].

teristics in user behavior:

- The working set of files, as observed over a period of days or weeks, must be small enough to fit on a portable's disk.
- It must be possible to predict the working set in advance, using hints such as the current working set, historical file access patterns [15], or known patterns in user behavior.

Analysis of the data we have collected shows that these characteristics are present in a number of different application domains.

2 Methodology

We collected our traces at Locus Computing Corporation, a software development and consulting firm, during the summer of 1993. One of Locus' products, PC/Interface (PCI) [8], is a DOS-to-UNIX file system implemented as a pseudo-disk driver on a DOS machine which communicates via Ethernet to a file server on the UNIX system, making the UNIX file system available to the DOS users as native PC files. In the environments monitored, the local DOS filesystem was used to store some applications software, but all shared corporate data was accessed via PCI. The UNIX server for PCI was modified to log opens, closes, and deletes of files. By avoiding read/write logging, we minimized the performance impact and kept the log files small. Log entries contain an operation type and subtype (e.g., open for read), the UNIX timestamp in seconds, the UNIX UID of the invoker, the process ID, the absolute pathname of the file, and the size of the file.

Three different user environments were monitored. In the first, referred to as "personal productivity," the server was a machine that acted as the network filesystem for 47 users running business-oriented applications such as e-mail, project and calendar scheduling, and word processing. These users did not tend to store important files on their own machines, so they generated high activity at the server. This server was traced for 1563 hours (65.1 days, or 9.3 weeks),[2] recording 4,637,924 accesses.

In the second environment, referred to as "programming," the server was a cluster of 10 machines running IBM's Transparent Computing Facility, an adaption of the Locus distributed operating system [12], which provides a single-system image to users of multiple machines. Each machine ran a separate PCI server, and logs from these servers were later combined for analysis. Most of the users of this server were programmers working on DOS-based software. Because they performed much of their work locally, accessing the shared server mostly to retrieve or update shared source files, they generated relatively little server activity. The traces on this server essentially reflect commits to a shared database, while omitting most localized file activity. This server was accessed by 64 users and was traced for 1693 hours (70.5 days, or 10.1 weeks), recording 93,719 accesses.

In the third environment, referred to as "commercial," the server was a single machine used by the accounting department to run a commercial accounting application. The master corporate accounting database was kept on the UNIX server, but all access to this (shared) database was via DOS workstations running the commercial package. This server was accessed by 7 users and was traced for 1257 hours (52.4 days, or 7.5 weeks), recording 371,830 accesses.

The nature of the traced environment (local files stored on PC's, with shared files stored remotely) parallels the expected behavior of mobile users, who will probably store heavily-used applications locally[3] but make extensive use of shared resources when they are network-connected. However, based on preliminary analysis of these traces, we also generated two modified traces that omitted certain characteristics we felt might be absent on portable platforms due to different software and user behaviors. For the commercial environment, we reduced all file sizes to a maximum of 1 MB, on the theory that very large databases would be represented by smaller slices in a portable environment. This change primarily affected the statistics on working-set sizes and the amount of data involved in *write conflicts* and *attention shifts*, which are measures of file sharing and working-set variability that we will define in Section 3. For the productivity environment, we eliminated all references to fax spooling and mail files, because such files are handled in a queued (as opposed to shared) manner in disconnected environments. This change affected all of the statistics we analyzed. These two data sets are referred to as the "reduced commercial" and "reduced productivity" environments in the tables and graphs.

Once the traces were collected, we canonicalized them using a simple `awk` script that converts relative pathnames to absolute form, correlates each close with the corresponding open and produces an output line whose format is independent of the operation type to make subsequent processing easier. These canonical-

[2] 50 days into this trace, there was a data gap of approximately 48 hours due to an administrative error. It does not appear that this gap affects the validity of the analysis.

[3] We hope that even these will eventually fall under the purview of an automated caching system.

ized files were then compressed and used as the basis for our analysis. The largest of these files (from the productivity server) is nearly 18 megabytes in its compressed form, and about 10 times that large when expanded.

Originally, we used a collection of shell and `awk` scripts for all analysis. As the collected data grew, many of these scripts became computationally impractical and were replaced by tailored programs. The current design performs the analysis in two phases. First, a single-pass program reads the data and extracts summary information of interest. For example, for each 24-hour day in the collected data, the extraction program writes a single line for each user giving the total size of that user's working set, measured in both megabytes and files. A second pass then analyzes these summary files with general-purpose statistical tools, generating the final tables and graphs presented in this paper.

3 Statistics

We generated the same statistics for each parameter in each environment: mean, standard deviation, and maximum. Besides the traditional measure of working-set size, we looked at two measures that have special application to mobility: *write conflicts* and *attention shifts*.

We define a write conflict event to occur when two users write to the same file within a relatively short time span. In a mobile environment, a conflicted file might be replicated on two or more computers, and the system would be required to automatically resolve these conflicts after the fact in a manner similar to the Ficus distributed file system [3, 4, 7, 13], to force the user to resolve them by hand [6], or to limit writing to only one user. We examined conflicting writes within a 24-hour period (corresponding to taking a machine home overnight) and a 7-day period (corresponding to traveling with a machine).

An attention shift occurs when a single user radically changes his or her working set. We identified attention shifts by looking at the working sets in successive active n-hour time periods (which did not necessarily represent adjacent days or weeks). Within each time period, we counted the total numbers of files accessed, k_1 and k_2, and then calculated $k = \min(k_1, k_2)$. Within the second period, we also counted the total number m of files that had not been referenced during the first period, but that had existed prior to either period.[4] An attention shift was defined to occur if $m \geq pk$, where $0 \leq p \leq 1$. Attention shifts can be characterized by the parameters p, expressed as a percentage, and n, the number of hours in the period. We use the notation $p\%/n$ to describe an attention shift parameter pair. Based on a sensitivity analysis (see Figures 6–8), we chose $p = 20\%$. We chose $n = 24$ and $n = 168$ (1 week) because these represent typical disconnection periods for many portable users.

A final characteristic of an attention shift is the *age* of the shift, which represents the amount of time which has elapsed since the user last referenced one of the "new" files. We estimated the age by locating the most recently-referenced "new" file (a file included in count m), and subtracting its reference time from the start time of the second period. This is a conservative measure, since it assumes that the most-recently-referenced file is representative of the entire group m of "new" files.

However, since many of the newly-referenced files did not appear previously in the trace, it was not always possible to find a file to use in calculating the age of the shift. In this case, we conservatively assumed that the "new" files had been referenced exactly one second before the beginning of the entire trace. Because of these two assumptions, the attention-shift ages reported in this paper are only a lower bound on the true ages that would be encountered by a predictive caching system.

The *bounded locality intervals* discussed in [9] are similar to attention shifts, but are parameterized on working-set sizes rather than on the expected length of a disconnection.

The statistics we report are:

Working-set statistics. For each day and week, we calculated the working set size in files, MB, and number of accesses. Means and standard deviations were calculated by averaging data across time for each UID, and then calculating the mean and standard deviation across the per-UID means.

Attention-shift statistics. For each 1-day and 7-day attention shift, we examined the total size of the working set needed to hold both the old and the new data (in files and MB). We also calculated the per-user attention shift rate per day and per week. Finally, we calculated the age of each shift.

Conflict statistics. For each conflict, we examined the number of users involved and the size of the file involved. We also calculated the per-user conflict rate per day and per week.

Success in mobile computing depends on small values for all of these statistics. Clearly, the working set must be small enough to fit comfortably on the typical portable's disk. The attention-shift rate should remain

[4] We eliminated files that were created during the second period because they are not problematical for a caching system that must predict which existing files need to be stored.

low, both so that the longer-period working set remains small and so that it is easier to predict the future working set based on recent behavior. The conflict rate must remain low to allow convenient file updates.

4 Analysis

The results of our analysis are very encouraging for our intended application, automated caching of files for mobile computers. As hoped, working sets are small and attention-shift rates are low. Conflict rates are generally low, and it is clear how one could handle conflicts in the environments that had high conflict rates. However, attention-shift ages tend to be high, indicating that a predictive caching system will need to exercise significant intelligence to ensure that a portable computer is prepared for attention shifts.

Each table of statistics given below lists the mean for the statistic, followed by the standard deviation (in parentheses) and the maximum. For example, in Table 1, the mean daily working set for the productivity environment was 1.0 MB, with a standard deviation of 2.0 MB and a maximum of 134.5 MB.

With the exception of Figures 6–8, all figures show the variation in a given measure over the duration of the trace. For example, Figure 1 shows the daily and weekly working sets for the productivity environment, for each day and each week captured during the trace.[5]

4.1 Working Sets

Table 1 summarizes the working-set sizes we observed. Figures 1–4 show the variation in mean and maximal working set sizes with time.

Mean working-set sizes tended to be small in all three environments, with the largest being about 18 MB per day and 24 MB per week, in the commercial environment. Maximal working sets were very large (148 MB per week) only in the personal-productivity environment, apparently due to a single `grep`-style operation that occurred in week 9. This "`grep` phenomenon" is clearly visible in Figure 1. Eliminating this single maximum produced a secondary maximum of only 76 MB. Maximal working sets in the other environments ranged only to 66 MB.

These working-set figures indicate that it will be easy to store enough files on a portable disk to satisfy the average user,[6] although some software or user behavior may have to change. (For example, instead of relying on a large `grep`, a user might use an inverted index to locate the files containing references to a particular string [10].)

4.2 Attention Shifts

Tables 2 and 3 summarize the attention shifts observed. Figures 6–8 show the sensitivity of attention-shift rates to the parameter p. Except in the commercial environment, the number of attention shifts steadily decreases with increasing p, but the exact shape of the curve is quite inconsistent. In the absence of a clear-cut change in curvature (a knee or cliff), to guide us in the selection of p, we chose $p = 20\%$, which is near enough to the peak of the curves that we will not tend to underestimate the number of attention shifts, yet not so small that we will detect a shift every time a user accesses one or two new files.

Figures 9–11 show the variations in attention-shift rates with time, for $p = 20\%$. The amount of data involved in attention shifts was generally small (33 MB or less), though the maxima were large (up to 152 MB; this follows from the size of the maximal working set and the definition of an attention shift). In all three environments, the number of attention shifts was surprisingly large and consistent, averaging up to 0.6 per user per week. This has serious implications for a predictive caching scheme, because it shows that simply caching least-recently-used files is not sufficient.

However, because of the small size of the working sets involved in the average attention shift, a well-designed predictive cache can afford to store both the old and the new set, so that attention shifts need not affect the usability of a mobile computer.

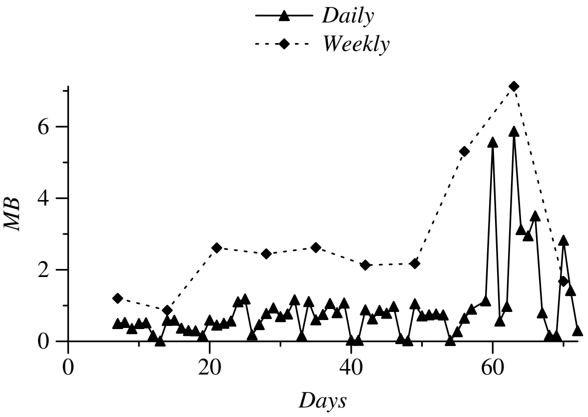
Of course, if there is space to store both the old and new working set, the question arises whether a simple LRU scheme would be sufficient to ensure that both working sets are available. The attention-shift age figures shown in Tables 2 and 3 belie this notion. For both the programming and the reduced productivity environments, the mean age of an attention shift is over 4 weeks and the maximum is near the length of the trace, indicating that an LRU cache would very likely have been flushed by transient phenomena before the older files were re-referenced. This hypothesis is strengthened by the observation that the conservative method of estimating the ages of previously-unreferenced files,

[5] In these and all other graphs, the lines connecting data points are present only to make it easier to see associated points, and are not meaningful in themselves. In particular, although the daily maxima in the right-hand sides of Figures 4 and 5 appear to exceed the weekly maxima, careful examination shows that only the connecting lines cross, and the actual data points for weekly maxima are always larger than the daily values.

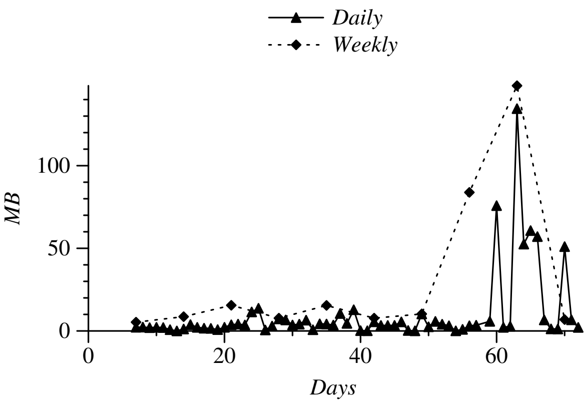
[6] We expect working-set sizes to change dramatically over the next few years as users move towards multimedia applications, but we also expect that disk sizes will increase sufficiently for portable computers to keep pace. In some sense, this phenomenon is self-regulating, since users will not tend to use images and sounds extensively if this would tax their portable storage capacity.

Environment	Daily WS Size (MB) *Mean*	*σ*	*Max*	Daily WS Size (Files) *Mean*	*σ*	*Max*	Weekly WS Size (MB) *Mean*	*σ*	*Max*	Weekly WS Size (Files) *Mean*	*σ*	*Max*
Productivity	1.0	(2.0)	134.5	39	(80)	3293	2.7	(4.7)	148.4	110	(215)	3284
Reduced Productivity	0.7	(1.8)	41.1	7	(10)	547	1.4	(2.8)	43.6	19	(31)	548
Programming	0.3	(0.4)	18.0	10	(27)	2153	0.6	(1.1)	18.3	22	(55)	2170
Commercial	18.2	(13.1)	65.0	294	(442)	1643	26.8	(16.6)	65.7	374	(553)	1638
Reduced Commercial	10.9	(6.0)	33.6	294	(442)	1643	16.8	(8.7)	33.8	374	(553)	1638

Table 1: Working-Set Statistics

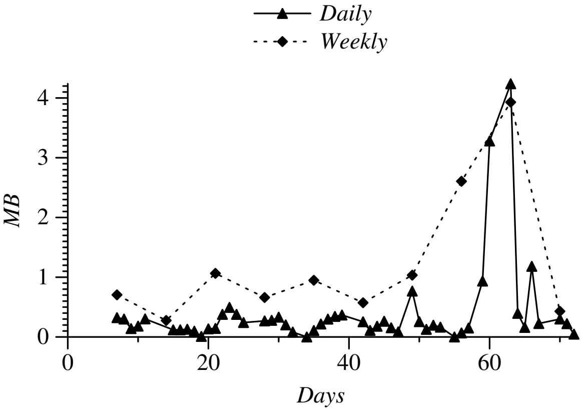


Average Productivity Working-Set Sizes

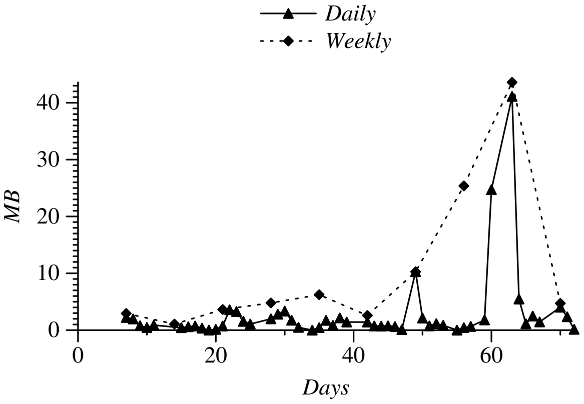


Maximum Productivity Working-Set Sizes

Figure 1: Working-Set Sizes for Productivity Environment

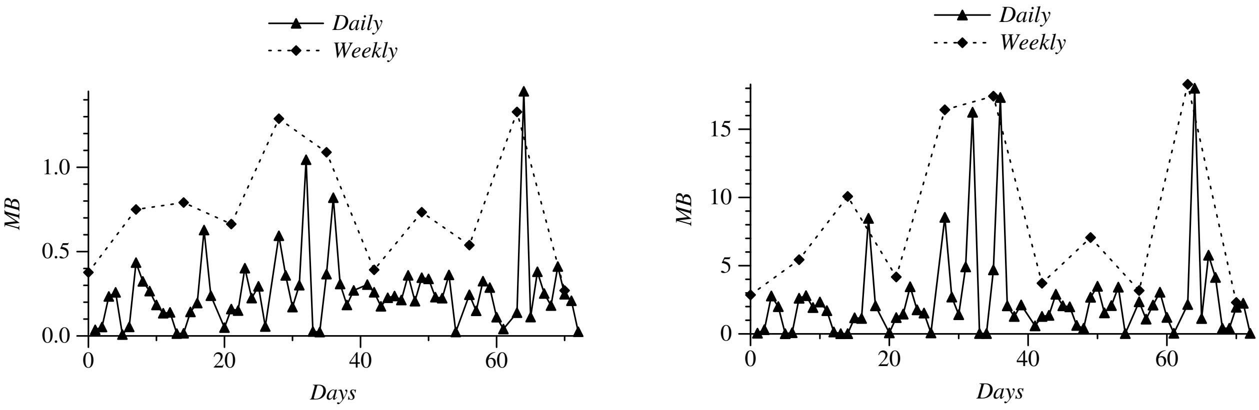


Average Reduced Productivity Working-Set Sizes



Maximum Reduced Productivity Working-Set Sizes

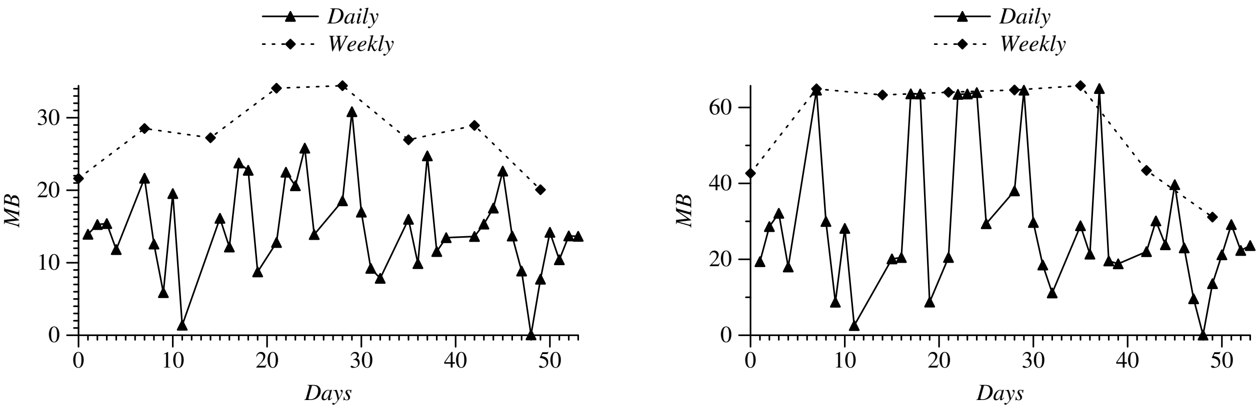
Figure 2: Working-Set Sizes for Reduced Productivity Environment



Average Programming Working-Set Sizes

Maximum Programming Working-Set Sizes

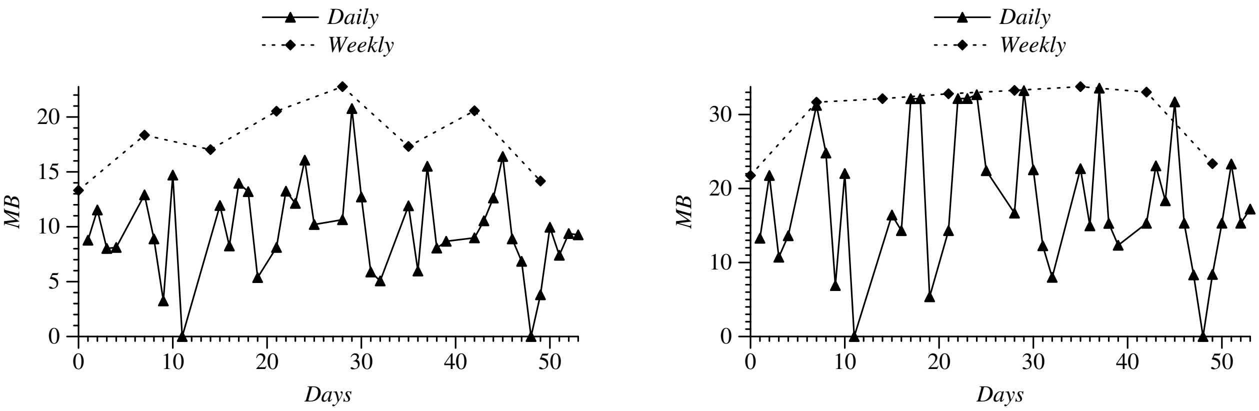
Figure 3: Working-Set Sizes for Programming Environment



Average Commercial Working-Set Sizes

Maximum Commercial Working-Set Sizes

Figure 4: Working-Set Sizes for Commercial Environment



Average Reduced Commercial Working-Set Sizes

Maximum Reduced Commercial Working-Set Sizes

Figure 5: Working-Set Sizes for Reduced Commercial Environment

	Number Per User Per Day			MB Involved			Files Involved			Age (Days)		
Environment	*Mean*	*σ*	*Max*	*Mean*	*σ*	*Max*	*Mean*	*σ*	*Max*	*Mean*	*σ*	*Max*
Productivity	0.4	(0.3)	0.8	1.6	(6.5)	135.7	64	(164)	3296	10.0	(15.7)	64.7
Reduced Productivity	0.2	(0.2)	0.5	0.8	(3.2)	41.1	13	(33)	548	26.2	(19.7)	64.7
Programming	0.3	(0.2)	0.5	0.6	(1.6)	20.9	16	(109)	2161	28.0	(21.3)	70.2
Commercial	0.3	(0.3)	0.9	21.8	(13.8)	65.7	217	(398)	1654	3.2	(4.6)	35.7
Reduced Commercial	0.3	(0.3)	0.9	14.6	(8.1)	33.8	217	(398)	1654	3.2	(4.6)	35.7

Table 2: 20%/24-Hour Attention Shifts (All Users)

	Number Per User Per Week			MB Involved			Files Involved			Age (Days)		
Environment	*Mean*	*σ*	*Max*	*Mean*	*σ*	*Max*	*Mean*	*σ*	*Max*	*Mean*	*σ*	*Max*
Productivity	0.6	(0.3)	0.8	4.7	(12.4)	151.8	177	(376)	3423	15.7	(15.2)	62.7
Reduced Productivity	0.3	(0.2)	0.4	2.0	(5.5)	44.3	37	(71)	553	32.4	(18.9)	62.7
Programming	0.4	(0.2)	0.6	1.7	(3.4)	22.6	55	(215)	2174	28.9	(20.0)	68.2
Commercial	0.5	(0.4)	1.0	33.3	(17.4)	66.8	420	(584)	1661	11.1	(6.1)	33.7
Reduced Commercial	0.5	(0.4)	1.0	21.1	(9.0)	33.8	420	(584)	1661	11.1	(6.1)	33.7

Table 3: 20%/168-Hour Attention Shifts

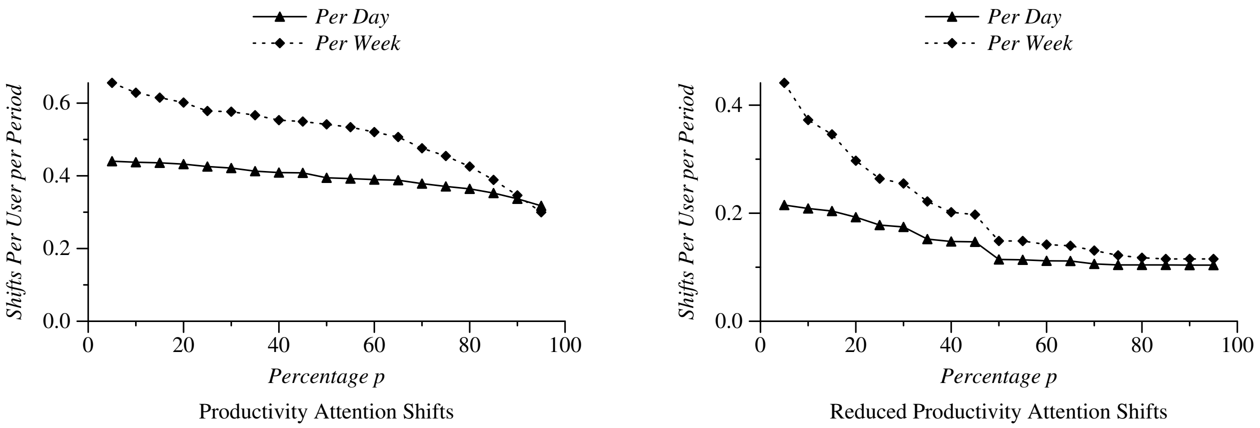
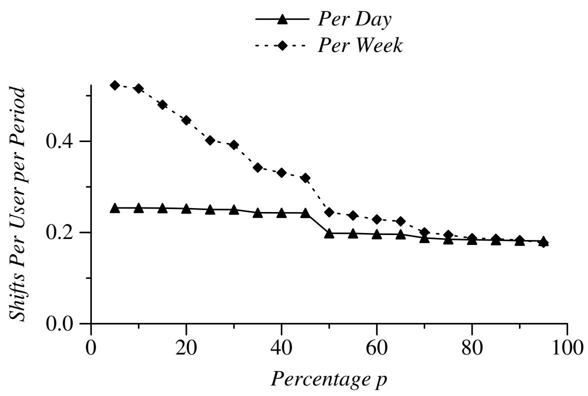


Figure 6: Attention-Shift Sensitivity for Productivity Environment



Programming Attention Shifts

Figure 7: Attention-Shift Sensitivity for Programming Environment

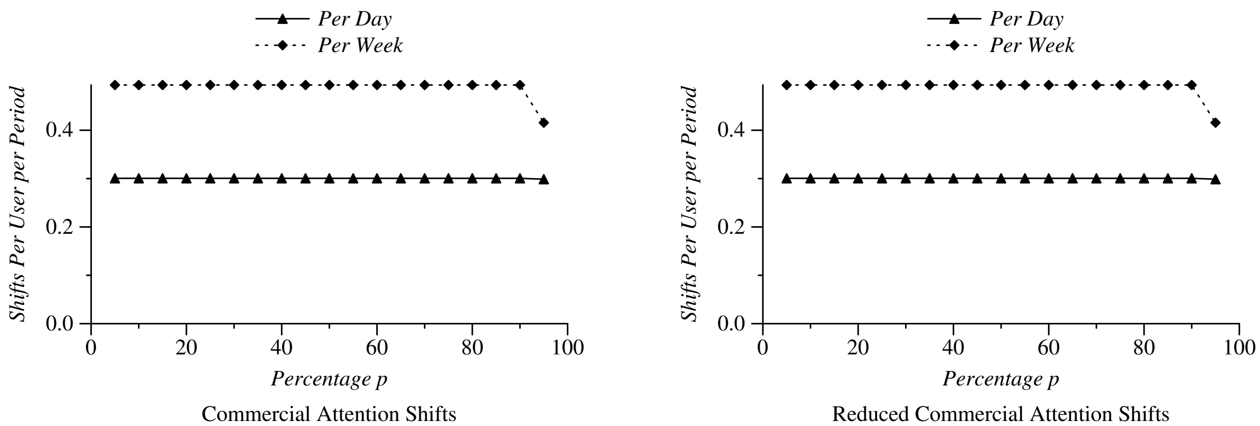


Figure 8: Attention-Shift Sensitivity for Commercial Environment

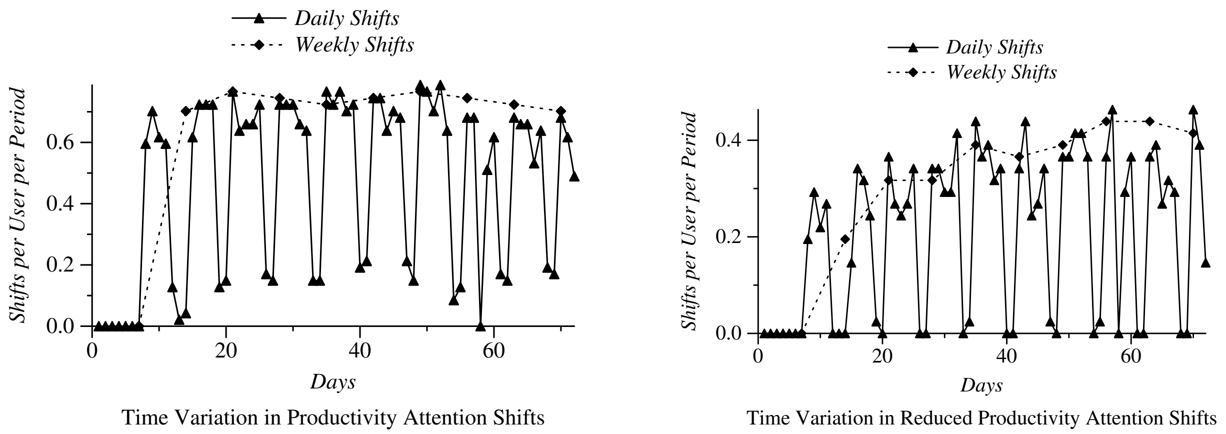
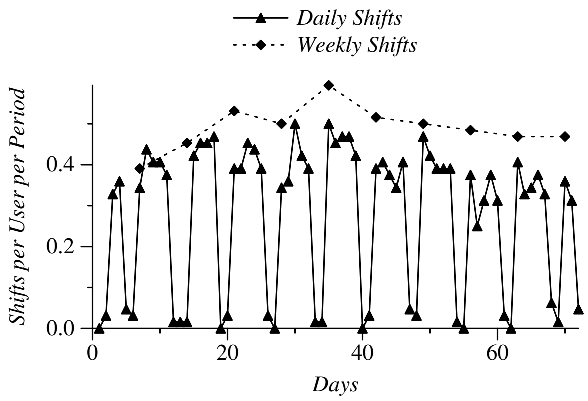
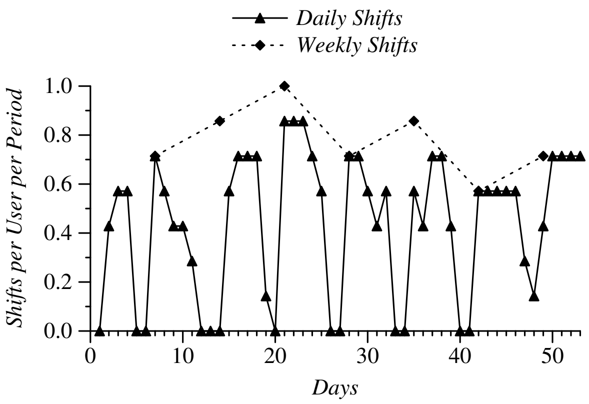


Figure 9: 20% Attention-Shift Rates for Productivity Environment

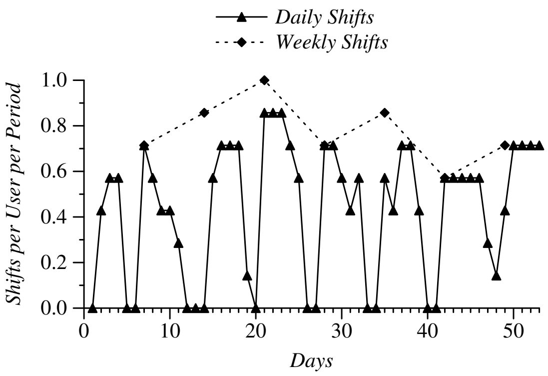


Time Variation in Programming Attention Shifts

Figure 10: 20% Attention-Shift Rates for Programming Environment



Time Variation in Commercial Attention Shifts



Time Variation in Reduced Commercial Attention Shifts

Figure 11: 20% Attention-Shift Rates for Commercial Environment

explained in section 3, would produce a mean age of approximately half the length of the trace (about 5 weeks) if there were absolutely no historical data in the trace. In actuality, the new working set may not have been accessed for many months and thus may have been flushed from even a very lengthy LRU cache. Other methods will be needed to ensure that a mobile machine will be prepared for an attention shift. The above data merely assures us that there will be room to store both today's and tomorrow's working sets once they have been identified.

4.3 Conflicts

Tables 4 and 5 show statistics about conflicts and their rate of occurrence, respectively. Figures 12–14 show the variations in conflict rates with time. Conflicts were very rare in the "programming" environment, averaging 0.01 conflict per user per day, and only 0.10 per week. In nearly every case only two users were involved in a given conflict, although occasionally a third would write to the same file within 24 hours.

As expected, the 7 users of the "commercial" environment, with its shared accounting database, produced a high conflict rate of 11 per user per week, with up to 6 users writing to the same file in a single day. In a mobile environment, an automated resolver similar to those discussed in [13] would be required to handle these numerous conflicts. Since accounting applications typically involve appending records to a transaction database, we expect that such a resolver would be easy to write.

The surprise was the "personal productivity" environment, which produced conflict rates up to 1.2 per user per day, with up to 22 users writing to the same file in a single 24-hour period. We examined these conflicts in more detail to discover the cause, and found that nearly all of them involved mailboxes or fax-spooling files.

Since both mailbox and spooling files operate in a modified append-only mode (all but one user appends to the end of the file, and a simple locking mechanism prevents update while other file contents are modified), this does not present a problem for mobility. In fact, the retry-on-failure queuing algorithm of mailers would handle mailbox conflicts with no software changes. In view of these observations, we generated the "reduced productivity" trace, which omitted these files from the statistics. With this change, the conflict rate dropped to only 0.04 per user per week, a number so small that it could conceivably be handled even without the help of automatic resolvers.

5 Future Work

Based on the above analysis, we expect to build a prototype caching system incorporating a prediction mechanism which, by observing user behavior, will calculate the current working set, detect attention shifts, and predict possible future working sets. A cache manager will then ensure that these working sets are available on the portable computer when it is disconnected from the network.

A cache miss during disconnection is a serious, often catastrophic event for a user who cannot continue to work in the absence of a critical file. There are only two real options for dealing with this case:

1. Provide enough alternate working sets that the user can shift to a secondary or tertiary task [6, 14].

2. Provide a foreground or background method that initiates communication (most likely expensive and slow) to retrieve the missing file [5].

We plan to provide both of these options in our prototype, though we hope to rely primarily on the first.

6 Conclusions

The data gathered and analysis performed in this study strongly indicate that predictive file caching for mobile computing is a feasible approach. However, the data also indicates that simple LRU caching is insufficient. Therefore, we conclude that more sophisticated automatic predictive file caching mechanisms will be required to make the file system of a mobile computer appear transparently the same as the file system of a desktop machine. We intend to investigate suitable algorithms for this purpose, guided by these results and by further analysis of our data.

Trademarks

PC/Interface is a trademark of Locus Computing Corporation. UNIX is a trademark of X/Open Company, Ltd.

References

[1] Mary G. Baker, John H. Hartman, Michael D. Kupfer, Ken W. Sherriff, and John K. Ousterhout. Measurements of a distributed file system. In *Proceedings of the Thirteenth Symposium on Operating Systems Principles*, pages 198–211. ACM, October 1991.

Environment	Daily Conflicts Mean	σ	Max	Weekly Conflicts Mean	σ	Max
Productivity	1.19	(1.16)	4.28	5.57	(3.58)	10.11
Reduced Productivity	0.00	(0.01)	0.05	0.02	(0.03)	0.07
Programming	0.01	(0.02)	0.06	0.10	(0.09)	0.28
Commercial	4.29	(4.74)	16.29	11.30	(8.92)	24.57
Reduced Commercial	4.29	(4.74)	16.29	11.30	(8.92)	24.57

Table 4: Conflict Rates

Environment	MB Involved in Daily Conflicts Mean	σ	Max	Users Involved in Daily Conflicts Mean	σ	Max	MB Involved in Weekly Conflicts Mean	σ	Max	Users Involved in Weekly Conflicts Mean	σ	Max
Productivity	0.02	(0.08)	2.05	3.39	(3.06)	22.00	0.02	(0.08)	2.05	3.61	(3.62)	27.00
Reduced Productivity	0.04	(0.04)	0.12	2.00	(0.00)	2.00	0.04	(0.04)	0.12	2.00	(0.00)	2.00
Programming	0.07	(0.16)	1.08	2.02	(0.15)	3.00	0.06	(0.12)	1.08	2.09	(0.29)	3.00
Commercial	0.22	(0.81)	5.37	3.16	(1.10)	6.00	0.27	(0.83)	5.37	3.16	(1.28)	6.00
Reduced Commercial	0.17	(0.81)	5.37	3.16	(1.10)	6.00	0.20	(0.83)	5.37	3.16	(1.28)	6.00

Table 5: Conflicts

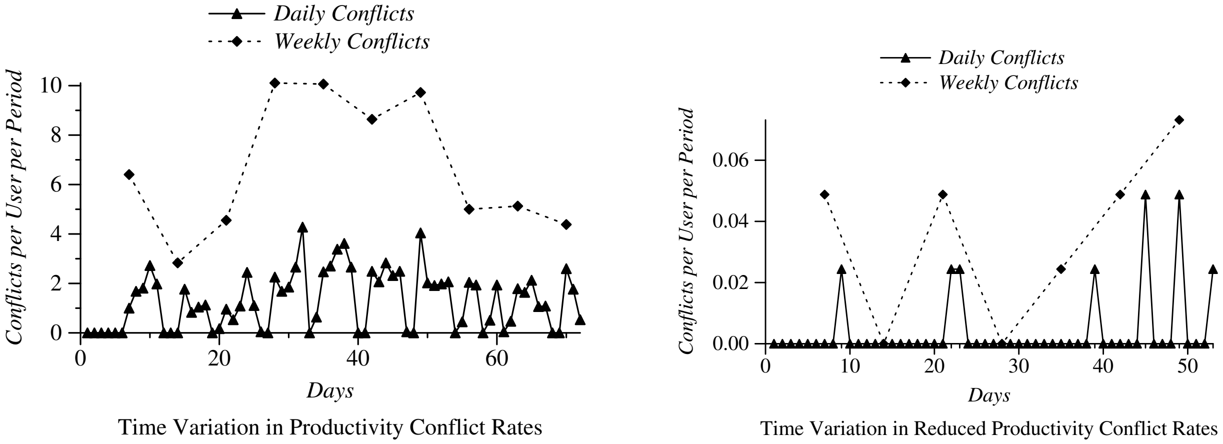
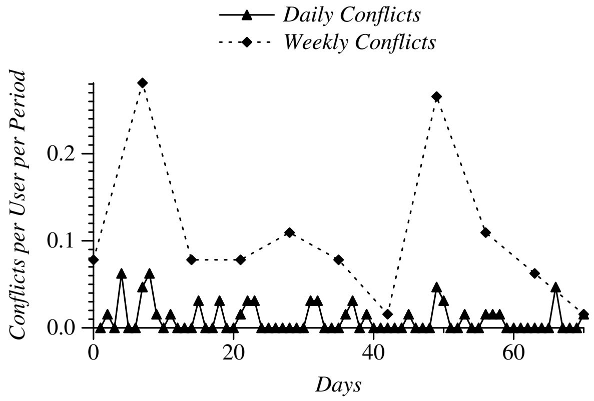
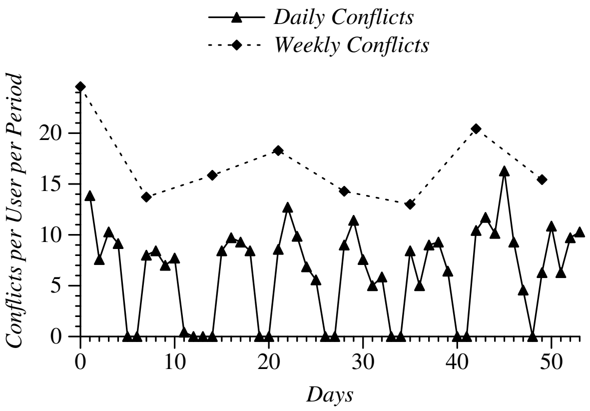


Figure 12: Conflict Rates for Productivity Environment

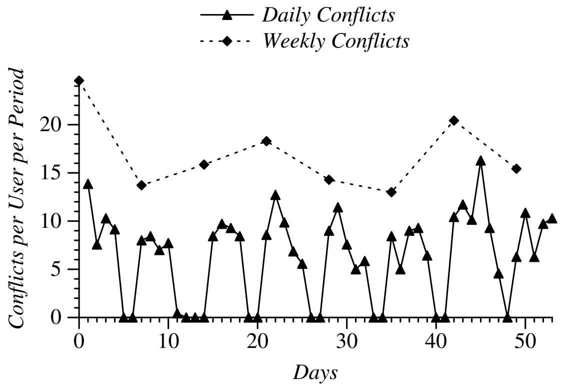


Time Variation in Programming Conflict Rates

Figure 13: Conflict Rates for Programming Environment



Time Variation in Commercial Conflict Rates



Time Variation in Reduced Commercial Conflict Rates

Figure 14: Conflict Rates for Commercial Environment

[2] Matthew Blaze and Rafael Alonso. Dynamic hierarchical caching for large-scale distributed file systems. In *Proceedings of the Twelfth International Conference on Distributed Computing Systems*, pages 521–528, June 1992.

[3] Richard G. Guy. *Ficus: A Very Large Scale Reliable Distributed File System*. Ph.D. dissertation, University of California, Los Angeles, June 1991. Also available as UCLA technical report CSD-910018.

[4] Richard G. Guy, John S. Heidemann, Wai Mak, Thomas W. Page, Jr., Gerald J. Popek, and Dieter Rothmeier. Implementation of the Ficus replicated file system. In *USENIX Conference Proceedings*, pages 63–71. USENIX, June 1990.

[5] L. B. Huston and Peter Honeyman. Disconnected operation for AFS. In *Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing*, pages 1–10. USENIX, 1993.

[6] James Jay Kistler. *Disconnected Operation in a Distributed File System*. Ph.D. dissertation, Carnegie-Mellon University, May 1993.

[7] Puneet Kumar and Mahadev Satyanarayanan. Supporting application-specific resolution in an optimistically replicated file system. In *Proceedings of the Fourth Workshop on Workstation Operating Systems*, pages 66–70, Napa, California, October 1993. IEEE.

[8] Locus Computing Corporation, Inglewood, California. *PC/Interface Reference Manual*, February 1993.

[9] Shikharesh Majumdar and Richard B. Bunt. Measurement and analysis of locality phases in file referencing behavior. In *Proceedings of Performance 86 and ACM Sigmetrics 86, Joint Conference on Computer Performance Modelling, Measurement and Evaluation*, pages 180–192, Raleigh, NC, May 1986. ACM.

[10] Udi Manber and Sun Wu. GLIMPSE: A tool to search through entire file systems. In *USENIX Conference Proceedings*, pages 23–32, San Francisco, CA, January 1994. USENIX.

[11] John K. Ousterhout, Hervé Da Costa, David Harrison, John A. Kunze, Mike Kupfer, and James G. Thompson. A trace-driven analysis of the UNIX 4.2 BSD file system. Technical Report UCB/CSD 85/230, UCB, 1985.

[12] Gerald J. Popek and Bruce J. Walker. *The Locus Distributed System Architecture*. The MIT Press, 1985.

[13] Peter Reiher, John S. Heidemann, David Ratner, Gregory Skinner, and Gerald J. Popek. Resolving file conflicts in the Ficus file system. In *USENIX Conference Proceedings*. USENIX, June 1994. To be published.

[14] Mahadev Satyanarayanan, James J. Kistler, Lily B. Mummert, Maria R. Ebling, Puneet Kumar, and Qi Lu. Experience with disconnected operation in a mobile computing environment. In *Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing*, pages 11–28, Cambridge, MA, August 1993. USENIX.

[15] Carl D. Tait and Dan Duchamp. Detection and exploitation of file working sets. In *Proceedings of the Eleventh International Conference on Distributed Computing Systems*, pages 2–9, 1991.

Authors

Geoffrey H. Kuenning is a Ph.D. candidate in computer science at UCLA. He received the B.S. and M.S. degrees in computer science from Michigan State University in 1973 and 1974, respectively. His research interests include operating systems, distributed environments, and mobile computing. He is a member of ACM, the IEEE Computer Society, and CPSR. His Internet address is `geoff@ficus.cs.ucla.edu`.

Peter Reiher received his B.S. in electrical engineering from the University of Notre Dame in 1979. He received his M.S. in computer science from UCLA in 1984, and his Ph.D. in computer science in 1987. He has worked on several distributed operating systems projects. His research interests include distributed operating systems, optimistic computation, and security for distributed systems. His Internet address is `reiher@ficus.cs.ucla.edu`.

Gerald J. Popek has been a Professor of Computer Science at UCLA since 1973. His academic background includes a doctorate in computer science from Harvard University. He co-authored "The LOCUS Distributed System Architecture," MIT Press, 1985, and has written more than 70 professional articles concerned with computer security, system software, and computer architectures. Dr. Popek is a principal founder of Locus Computing Corporation, the largest independent developer of UNIX-based connectivity and distributed processing software technology.