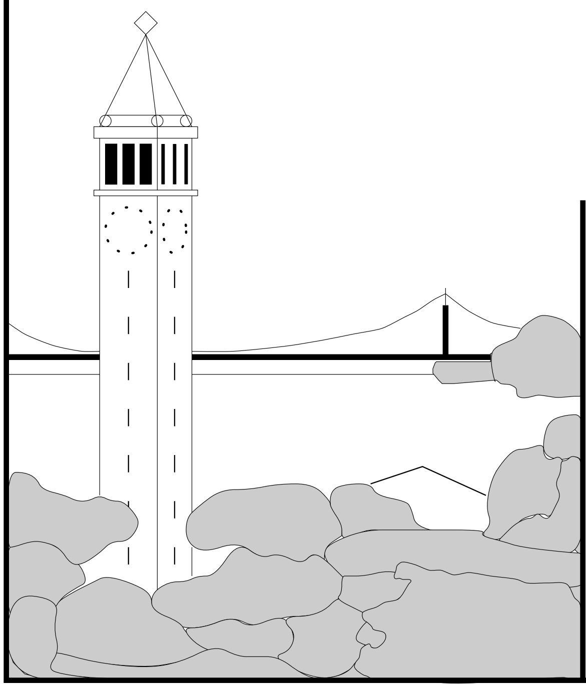
Analysis of Personal Computer Workloads

Min Zhou and Alan Jay Smith



Report No. UCB/CSD-99-1038

January 1999

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Analysis of Personal Computer Workloads *

Min Zhou and Alan Jay Smith
Computer Science Division, University of California, Berkeley, CA 94720-1776

January 1999

Abstract

Most published research on system behavior and workload characterization has been based on either Unix systems or large, usually IBM, mainframe systems. It is reasonable to believe that user behaviors and workloads are different for PC systems. Further, the aspects of system design and most needing study have changed from the mainframes dominant in the 1960s and 1970s, and the Unix systems that became so popular in the 1980s to the PCs that seem to be rapidly taking over many or most aspects of computing. Our analysis focuses instead on Windows95, which is currently the most widely used computer operating system; the newly released Windows98 is very similar. In this paper, we describe our workload analysis based on 36 sets of traces collected from Intel Pentium based PCs running the Microsoft Windows95 operating system. Following the discussion of our Windows95 trace data, we present some descriptive and statistical characterization of this data, directed principally at user behavior and file system behavior.

1 Introduction

All aspects of computer system design and optimization depend strongly on knowledge of or assumptions about the workload that the system is intended to or does support. Multi-user timesharing computer systems such as UNIX systems and mainframe computer systems have been extensively studied. Much comprehensive system workload analysis has been done on these systems; e.g. [Smit81][Oust85][Bake91]. There is far less data and analysis available for personal computer workloads, the subject of this paper. We will focus on two major aspects of computer system workload in our analysis: user input behavior and file system workload characteristics. Since the most dominant PC operating system and architecture are Microsoft Windows95 and Intel x86 based Architecture (IA) respectively, we've selected Intel based PCs running Windows95 to study. We will refer to Intel based PCs running the Microsoft Windows95 operating system as "PC systems" in the rest of this paper. We note that Windows98 has been recently released, and is very similar to Windows95. We would expect that our characterization of the Windows95 workload should also apply to the Windows98 environment.

The first step in this research project was to design and develop a Windows95 PC system tracer, WMonitor, which collects traces of user and file system activities. The design and operation of that tracer is described in [Zhou98]. The original reason for this tracing project was to collect data to be used for studies of power management in portable computers. Accordingly, our trace collects records of user and file system activity. Details of the record types, formats and semantics appear in [Zhou98]; we provide more limited information as necessary in this paper.

The selection of users to trace has a significant impact on the characteristics of the workload collected. In this paper we report on analysis based on traces collected from 36 real users, including engineers, scientists, managers, home users and school students, using a variety of system configurations. In a few cases, we've broken out our analysis by user type.

The analysis presented in this paper includes an overall summary, and then focuses on specific topics related to user input behavior and file system operation. Our user input behavior study covers such major user input characteristics as commonly used applications, user idle periods and command

*The authors' research has been supported in part and at various times by the National Science Foundation under grants MIP-9116578 and CCR-9117028, by NASA under grant NCC 2-550, by the State of California under the MICRO program, and by Microsoft, Sun Microsystems, Fujitsu Microelectronics, Sony Research Laboratories, Microsoft Corporation, Cirrus Logic, Quantum Corporation, Toshiba Corporation, and Intel Corporation.

clustering, etc. The file system workload study considers the distribution of different file system calls, file system IO throughput, file system idle periods, file read/write size distributions and file size distributions. We also compare our analysis to some existing file system analysis results based on UNIX file systems[Oust85][Bake91][Cost85] and mainframe computer file systems[Smit81].

The rest of this paper is organized as follows: Section 2 briefly discusses some previous related research. Section 3 is a brief summary of our tracer and trace data; we also provide information about the users and systems being traced. Section 4 provides some general characterization of the traces analyzed in this paper. Section 5 analyzes user input behavior, and Section 6 characterizes the system workload from the file system perspective. Section 7 provides some miscellaneous analysis results and discusses the limitations of our analysis. Finally, section 8 summarizes our results and discusses directions for future work. In the appendices we provide a description of our tracer and trace formats, a system overview of Windows95, and some more detailed analysis, as shown in a few additional figures and tables.

2 Related Work

In this section, we briefly reference some related workload studies. We concentrate on related work by these others and others at Berkeley, but cite some other work as well; this is a small fraction of the dozens to hundreds of trace based workload studies.

Ousterhout et al[Oust85][Bake91] studied the Sprite distributed file system. They measured application-level file access patterns, throughput and distributed system file caching behavior. Da Costa[Cost85] discussed a BSD 4.2 UNIX file system tracing package and presented some summary statistics; see also[Zhou85]. Among the topics considered were general file system activity statistics, file IO transfer size and duration, and IO buffer allocation.

Kotz et al[Kotz95] traced and characterized the file system workload on a scientific multiprocessor system. Spasojevic et al[Spas96] studied the performance and workload characteristics of a wide-area distributed file system. They profiled wide-area file system storage capacity, volume activity, client-server interaction behaviors, cache performance and availability. Smith[Smit81] studied long term file access patterns on a mainframe computer system.

Becker et al[Beck91] analyzed UNIX paging behavior and concluded that paging activity accounts for 15% to 21% of all disk block accesses. Our analysis yields a lower result for this statistic; our results are also lower than the figures shown in[Oust85] and[Bake91]. Zivkov et al [Zivk97a] [Zivk97b] studied DB2 disk reference patterns and disk caching. Ruemmler[Ruem93] described disk access patterns on HP UNIX workstations. His study found that majority of all disk operations are writes (56% to 58%), disk accesses are rarely sequential, and the majority of disk accesses are resulted from non-user data accesses(swapping, metadat and program execution).

Nolan et al[Nola74] presented a workload characterization based on the user workload collected from Xerox Sigma timesharing systems. Hanson et al[Hans85] studied user input in the UNIX shell command environment and characterized UNIX shell usage.

Douglis et al[Doug94] studied the file system level disk activities of Apple Powerbook computers. Lorch et al[Lorc97] profiled the system resource usage on Power-PC systems. Li et al[Li94] studied file system level disk activities of DOS/Windows-3.1. Chen et al[Chen96] presented a comparative performance analysis of Windows31, WindowsNT and NetBSD based on their Pentium performance measurements. Their study indicates a significant part of the cost of system functionality in Windows systems is due to the OS structures rather than the API required by Windows applications. Lee et al[Lee98] traced and characterized several Windows applications under Windows NT on the x86 processor. They used a binary instrumentation engine, Etch, for the x86-Windows NT in their trace collection. Instruction set level desktop application performance was studied from the perspectives of computer architecture. These desktop applications were contrasted to the programs in the integer SPEC95 benchmark suite.

3 Trace Description

In this section we discuss the type of data used in our analysis and the users and machines from which the trace data were collected; further information about the tracing can be found in the appendices. In Appendix I, we provide an overview of the Windows95 operating system and those application programs shown in our study. We explain how we collected our trace data from Windows95 in Appendix II.

3.1 PC Workload Traces

Our PC workload traces were collected during our earlier related tracing project.[Zhou98] The traces were collected with three end-uses in mind. First, we are studying power management in portable computer systems (see e.g.[Lorc97]), and we wanted to collect those activities reflecting certain aspects of power consumption-user activity and disk activity. Second, we are interested in extending some of our previous studies in disk caching[Zivk96][Smit85] to PC-type systems. Third, we are also interested in generally characterizing the PC workload, which is the major focus of the work described in this paper.

Our expectation was that the workload we would observe on the PCs will differ from previously studied systems: the operations of personal computer systems are more tightly coupled with user activities (e.g. there are almost no batch or background jobs in PC workloads); the PC workload is more bursty because of single user and single application type workload, and the use of a GUI; Windows95 sometimes does not behave in an optimized way because it was designed to support both 32-bit Windows applications and old MS-DOS programs as well as 16-bit Windows applications.

Our traces are actually two separate trace streams: user activity traces and file system traces. User activity traces consist of 1) user keyboard input traces, 2) user mouse input traces, and 3) traces of user-input-focused windows, which we used for deriving application activities. The user-input-focused window is the window where the user mouse inputs and keyboard inputs are accepted. Our file system traces are generated from logical file system accesses and include function call names, file names, file handles, logical addresses, block sizes, etc. Since the virtual memory paging and swapping of Windows95 is implemented on top of the file system, our file system traces also contain virtual memory paging and swapping information.

Each trace, over a continuous period of time, is collected as a file. The following file is a WMonitor trace record file example:

```
USER_ID: 761
StartTime: 9E9C4F
Time  Type  Funct   Details
130   3     OPEN    C: [223] \DAT\WMONITOR.INI
0     3     WRITE   C: [223] 93
0     3     CLOSE   C: [223]
0     3     SEEK    C: [2A8] 4B400:B
0     3     READ    C: [2A8] 200
A     2     [e54]   C:\WMONITOR\BIN\WMONITOR.EXE
0     3     READ    C: [271] 1000 MM
0     3     FATTR   C: \WINDOWS\SYSTEM\MFC40LOC.DLL
0     3     FDOPN   C: \WMONITOR\BIN\*.*
DB    0     K_DN    11
96    0     K_UP    91
0     3     RENAM   C: \DAT\DATA.ZIP \RECYCLED\DCO.ZIP
0     3     DIR     C: QLGD \WMONITOR\BIN\MSGHK.DLL
0     3     DELET   C: \RECYCLED\DESKTOP.INI
0     1     START_MV
9E    1     STOP_MV
... ...
Stop_Time: 1D41D3C
```

There are four types of trace record events: keystrokes, mouse events, file system calls and active application switches. Each trace event was time-stamped incrementally by the reading number from a Windows95 internal millisecond counter which keeps track of the elapsed time since Windows was last started. A keystroke trace record indicates one of the keys on the keyboard changing status (key, and up or down). Thus note that a normal keystroke is two events, a capital letter is four events (regular key and shift key, both up/down), and special characters or functions (e.g. alt-cntl-delete) can be even more events. A mouse trace record indicates the mouse device event (a button click or a movement).

An application trace record includes the name of the new active application and the active window handle. Application traces are referred as window (user-input-focused window) switch traces in the rest of this paper. Note that we treat window switch events as application switch events because in a graphical user interface (GUI) environment, an application switch event usually coincides with a window switch. A file trace record contains the file system call type and other file system call control information such as disk drive names, file handles, file names, bytes transferred, etc. A file trace record can be of different formats depending on which file system call function. Details of the tracer and trace formats are in Appendix II.

3.2 Machines and Users Studied

There are many different types of PC users, ranging from game players to engineers, and they may differ in their workload profiles. Laptop PC users may also behave differently from Desktop PC users. Thus it is very difficult to define or select "typical" PC users or to collect a "typical" PC workload. We have attempted to collect as large a number of user traces as possible, over as wide a range of user types and machine types, including both laptop and desktop PCs. The users being traced include engineers, managers, assistants, students, home PC users, and some others. The workload being traced includes software development, computer aided design, logic synthesis and simulation, document writing, Web browsing, remote-dialup, PC game playing, etc.

Our Windows95 traces used in the paper were collected from a few home PCs and a number of industry PCs in several corporate sites including Intel Corp., Quantum Corp., Sony Corp., Toshiba Corp., and Fujitsu Corp, each of which has funded this research at some time. Most of our trace data was collected in 1997. Thirty-six sets of traces are discussed here. Each set of traces was collected from a separate PC machine/user over the period of a few days to a few weeks. Since the portion of the time that each machine was powered on varied a great deal, our tracing time for each user also varies widely.

Table 1 shows some characteristics for each machine and user traced.

4 Workload Overall Statistics

Here we discuss the overall PC workload statistics over our trace data sample. Table 2 shows the trace size. "Number of Trace Files" is the total number of trace data files which are used in our analysis. As shown in the table, the average size of compressed traces per user is about 343.6 / 36 = 9.5M bytes. The tracing time is defined as the duration during which the traced PC was powered on and the tracer tool was enabled. The average tracing time per user is 3092 / 36 = 85.9 hours. In the same table, we show the fraction of time that the user status can be assigned to the categories of: busy, active, thinking, and inactive. We define a trace period as an "idle" period if no trace event happened within this period of time. We define "busy" time as a period of the trace during which there was no idle period longer than 0.5 second. We define "active" time as the duration of all the idle periods each longer than 0.5 second and shorter than 5 seconds. We define "thinking" time as the duration of all the idle periods each longer than 5 seconds and shorter than 5 minutes. We define "inactive" time as the duration of all the idle periods each longer than 5 minutes. We can see that "thinking time" accounts for a significant portion of the entire tracing time. This classification is useful for studies of power management, since various system components are typically turned off after certain periods of inactivity. For comparison, we also show the total "tracing-off" time in the same table. Tracing-off time is the total tracing calendar time minus the total tracing time. Tracing-off time covers the period when either the tracing target system was powered off, or the tracer was disabled by the user.

Category	Statistics
Number of Users	36
Number of Trace Files	1894
Total Data Size (compressed data size)	4801 M bytes 343.6 M bytes
Total Records	184,095,396
Total Tracing Time -BusyTime (idle<0.5s) -ActiveTime (0.5s<idle<5s) -ThinkingTime (5s<idle<5m) -InactiveTime (idle>5m)	3,092.0 hours 438.3 hours 960.4 hours 1245.5 hours 448.8 hours
Tracing-Off Time	9860.8 hours

Table 2: Trace Data Overall Statistics for 36 Traces

Table 3 shows the number and rate of trace events. The data shown in the table are the arithmetic mean values over the 36 trace sets. (I.e. the averages for each trace are then averaged.) File system trace records account for 94.4 percent of the total number of trace records. "KeyRec", "MouseRec", "WinRec", "FSysRec" and "VMRec" in the table represent keyboard trace record, mouse trace record, window switch trace record, file system trace record, and Windows95 virtual memory file system call trace record, respectively. In this table and the rest of the paper, we will use the term "file system calls" for simplicity whenever we discuss the regular file system calls, but virtual memory operations and tracer file system calls are not included. The PC users input by means of a mouse device as frequently as by a keyboard. The PC users switch from one user-input-focused window, i.e. a foreground Windows process, to another window as often as about once per minute. File system calls invoked by virtual memory activities, including paging and memory swapping, account for only a small part of all file system activities.

Figure 1 shows the busy period distribution as a function of the longest idle interval contained within the busy period. Three busy periods are shown: that for the user (input) only, for the file system only, and for the system as a while. The Y axis is the cumulative fraction of busy time over the total tracing time, given a certain idle time length. For example, consider the case in which

Number	Brand	Model	Type	Mem	Disk (C:/D:/E:)	Cal-TM	TrcTM	Ratio	TrcEvent	Corp.	User-Type
1	Toshiba	Protégé-610	laptop	16M	687M	199.5 h	30.96 h	16%	1116999	Intel	Engineer/Hdware.
2	Toshiba	Protégé-610	laptop	16M	687M	1028.2 h	54.77 h	05%	1990876	Other	HomeUser/Pilot
3	Digital	HiNote-UltraII	laptop	64M	1372M	344.2 h	116.47 h	34%	8122170	Fujitsu	Director
4	Fujitsu	Lifebook-v655tx	laptop	48M	1293M	503.5 h	153.50 h	30%	4753461	Fujitsu	Manager
5	Fujitsu	Lifebook-v655tx	laptop	48M	1293M	719.3 h	195.21 h	27%	4619375	Fujitsu	Manager
6	Fujitsu	Lifebook-v655tx	laptop	48M	1293M	185.1 h	36.74 h	20%	654949	Fujitsu	Manager
7	Fujitsu	Lifebook-v655tx	laptop	48M	1293M	201.5 h	46.16 h	23%	1129639	Fujitsu	Engineer/IC
8	Fujitsu	Lifebook-v655tx	laptop	48M	1293M	215.5 h	60.97 h	28%	605928	Fujitsu	Engineer/Docmnt.
9	Fujitsu	Lifebook-v655tx	laptop	48M	1293M	715.1 h	179.68 h	25%	4188428	Fujitsu	Engineer/Cad
10	Toshiba	Pentium-PC	laptop	24M	500M	215.3 h	15.13 h	07%	165920	Toshiba	Engineer/IC
11	Toshiba	Satellite-110ct	laptop	24M	775M	215.0 h	36.93 h	17%	613989	Toshiba	Engineer/Docmnt.
12	IBM	Thinkpad 760ed	laptop	32M	1.2G	535.5 h	64.98 h	12%	6745879	Ahold	Manager
13	Dell	Latitude CP233	laptop	64M	3.8G	301.4 h	80.69 h	27%	10116647	3com	Engineer/Docmnt
14	Winbook	XL233	laptop	32M	3.0G	125.6 h	23.35 h	19%	2728398	Quantum	Manager
15	IBM	Thinkpad 560	laptop	40M	2G	316.5 h	52.6 h	17%	17921936	Quantum	Marketing
16	Dell	Optiplex GXpro	desktop	96M	2G/1G	510.0 h	77.80 h	15%	14371370	Quantum	Manager
17	Dell	Optiplex GXpro	desktop	96M	2G/1G	170.2 h	27.62 h	16%	9004396	Quantum	Web-Master
18	PC	Pentium-266	desktop	64M	6.4G/3.1G	331.5 h	146.02 h	44%	13924018	Other	Consultant
19	PC	Pentium-120	desktop	24M	500M/4G	128.5 h	21.22 h	17%	1216834	Intel	HomeUser/Engnr
20	PC	Pentium-90	desktop	32M	1.2G	86.3 h	19.63 h	23%	648849	Intel	Researcher
21	Dell	Dimention-133	desktop	32M	1547M	972.4 h	81.91 h	08%	1447485	Other	HomeUser/Studnt.
22	Compaq	Prolinea-5150	desktop	16M	2G	430.5 h	40.28 h	09%	3648197	Sony	Engineer/Sftware.
23	Sony	PCV-120	desktop	64M	2G/2G	374.2 h	27.82 h	07%	2603819	Sony	Engineer/Sftware.
24	Sony	PCV-120	desktop	64M	2G/2G	438.6 h	120.38 h	27%	4246332	Sony	Engineer/Sftware.
25	AST	MS-T 5166	desktop	64M	2G/2G/1G	459.7 h	51.60 h	11%	9415271	Sony	Engineer/Sftware.
26	Gtw2k	P5-166	desktop	64M	1.5G	377.8 h	307.98 h	82%	9475388	Sony	Engineer/Hdware.
27	Sony	PCV-120	desktop	32M	2G	380.3 h	352.98 h	93%	10053795	Sony	Engineer/Video
28	Sony	PCV-120	desktop	32M	2G	378.7 h	100.96 h	27%	2041658	Sony	Manager
29	Sony	P55c	desktop	32M	2G/2G/1.6G	191.8 h	56.90 h	30%	9166259	Sony	Engineer/Sftware.
30	Toshiba	Pentium-PC	desktop	24M	500M/500M	216.1 h	52.30 h	24%	5240669	Toshiba	Assistent
31	Toshiba	Pentium-PC	desktop	48M	500M/500M	230.8 h	29.96 h	13%	2482814	Toshiba	Engineer/CAD
32	Toshiba	Pentium-PC	desktop	64M	1.2G	215.6 h	46.27 h	21%	954125	Toshiba	Engineer/Progmer
33	Toshiba	Pentium-PC	desktop	48M	1G	238.1 h	59.49 h	25%	1776593	Toshiba	Engineer/Progmer
34	Toshiba	Pentium-PC	desktop	24M	500M/1.5G	188.6 h	42.02 h	22%	5332521	Toshiba	Engineer/Docmnt
35	Toshiba	Pentium-PC	desktop	48M	1.2G	596.1 h	49.17 h	08%	8067977	Toshiba	Engineer/CAD
36	Toshiba	Pentium-PC	desktop	48M	500M/500M	216.8 h	56.82 h	26%	2316542	Toshiba	Clerk
Average (arithmetic mean)						359.8h	81.0 h	23.8%	5080820		
Standard deviation						223.6	71.1	0.178	4504590		

Table 1: Profile Data for Machines and Users Traced (We show both calendar time ("Cal-TM") and tracing time ("TrcTM") in the table. Our calendar time for each user/machine is measured by the number of hours between the date/time of the first record and that of the last record. Our tracing time for each user/machine is measured by the number of hours when the machine was powered on and the tracer was enabled. "Ratio" is the ratio of tracing time to calendar time. "TrcEvent" in the table is the total number of trace events for each user/machine being traced. Averages (arithmetic mean) and standard deviations are also shown, as appropriate.)

	Record#	Percentage	Rec#/h
KeyRec	3688020	2.05%	1192.7
MouseRec	3345516	1.86%	1082.0
WinRec	167184	0.09%	54.1
FSysRec	169528248	94.44%	54827.3
VMRec	2776680	1.55%	898.0
Total	179505648	100%	58054.1

Table 3: Trace Event Type Statistics ("Record#" is the number of one type of trace records, "Rec#/h" is the number of trace records per tracing hour.)

the user/file_system/overall_system is defined to be busy when when no user/file_system/overall_system idle period longer than 400 seconds (x=400) occurs, then the cumulative user input busy time is about 50%, the cumulative file system busy time is about 81%, and the cumulative overall system busy time is about 83%, respectively. As may be seen, user idle periods are more frequent and longer. Since the file system trace records constitute more than 90% of all trace records, and since the file system idle pattern has fewer file system idle periods with long idle length, the overall system idle pattern is very close to the file system idle pattern. The steps on the curves in the figure are caused to some extent by automatic periodic events in the system; these are discussed later in this paper.

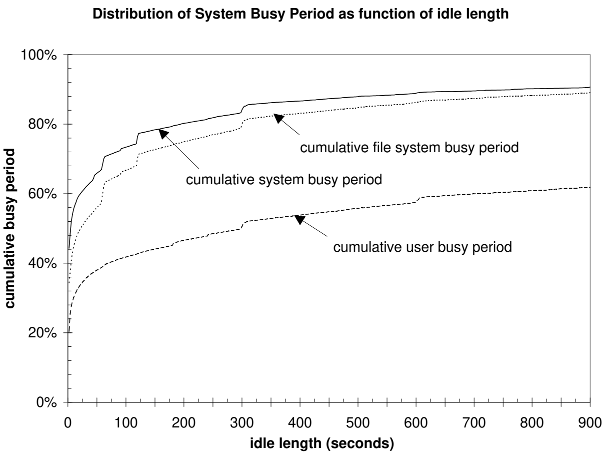


Figure 1: System Busy Time Distribution as Function of Maximum Idle Length

From the window switch traces, we are able to determine the names of the most frequently accessed Windows applications. Table 4 lists some of the application examples. We can roughly divide these applications into six categories, which are listed in the same table.

Category	APPLICATION (exe,dll)
Windows System	EXPLORER, SHELL32, COMDLG32, WINHELP, MPRSERV, ...
MS-Office/ Group-Ware	WINWORD, ACCESS, POWERPNT, EXCEL, COREL70, ...
Engineering Tool	MSDEV, DDRAW, ...
Misc. Tool	NOTEPAD, CALC, PHOTOSHOP, WINZIP, ACRORD32, ...
Internet Browser	NETSCAPE, IEXPLORE
Dos Application	MASM, QUICKEN, ...

Table 4: Traced Applications and Categories (application files in the table are in the format of "exe" or "dll")

We estimate which application is active by determining which user-input-focused windows is active. Note that this is only an approximation. First, we do not know the exact time an application was started and ended; the running time of an application is estimated by the difference in the times between when the window was entered (including opened) and exited (including closed). The number of times that an application is invoked is estimated by the number of times that the associated user-input-focused window was switched to. We also assume that all trace events were contributed by the application which had the user-input-focused window. For a single user PC system, we believe that these are reasonable approximations, but as will be seen later, some anomalies in the data do occur.

The trace event frequencies vary from one application to another. For example, some have very frequent user inputs, while some do many IOs. For the most frequent of the over 2000 different applications observed in our 36 trace sets, we will provide some simple statistics. For each application listed in Table 5, there is a brief description in Appendix I.

Table 5 shows information about 30 of the most frequently run applications ("Application"). These were selected based on the fraction of time that each application was traced to be running ("Time"). In the same table, we also show the average number of times each application was invoked per hour ("Invoked"), the rank numbers ("(r)") for the top 15 applications – ranked by the frequency of invocation, the average number of user keyboard/mouse inputs per hour ("KeyEvnt"/"MouseEvnt") while this application was running, the average number of file system calls per hour ("FSCall") for this application, and the average number of virtual memory system file system calls per hour ("VMFSCall"). These numbers are the averages over the 36 sets of traces. The total of the column "Time" is 100%, and the overall numbers for columns "Invoked", "KeyEvnt", "MouseEvnt", "FSCall", and "VMFSCall" for all applications are shown in Table 3.

Please note that there can be multiple active applications running, and the switching of user-input-focused windows may not exactly match the switching of applications. Thus there exist some anomalies in the table, such as the non-zero user keyboard inputs for the SCREEN-SAVER application. Also note that the number of mouse input events can be affected by the non-user mouse movement events generated occasionally by the Windows95 system.

A further breakdown of some of these statistics for these applications, by user type, appears in Appendix III.

A#	Application	Time	Invoked(r)	KeyEvnt	MouseEvnt	FSCall	VMFSCall
1	SCREEN-SAVER	22.27%	0.631 (15)	0.493	69.740	81851.281	91.126
2	MSDOS-PROMPT	11.90%	20.795 (1)	2287.842	2403.032	38302.695	1146.286
3	EXPLORER.EXE	8.75%	9.766 (2)	360.204	1178.353	56780.555	1905.136
4	WINWORD.EXE	7.27%	2.748 (4)	3768.675	1828.820	89175.586	3355.310
5	NETSCAPE.EXE	5.42%	1.914 (7)	894.104	1717.102	98169.844	1762.945
6	SHDOCVW.DLL	3.73%	2.964 (3)	608.463	2689.685	110360.680	4216.126
7	EUDORA.EXE	4.92%	0.875 (12)	736.941	643.741	26085.711	78.857
8	XVISION.EXE	3.86%	0.523	7694.424	226.582	57960.309	319.893
9	MSDEV.EXE	3.44%	1.671 (8)	3131.524	1251.985	69845.750	3572.495
10	EXCEL.EXE	2.52%	2.114 (5)	2351.149	3246.628	34757.672	1696.109
11	OUTLLIB.DLL	2.14%	1.066 (10)	2909.180	1066.946	148826.953	615.960
12	POWERPNT.EXE	1.92%	0.613	1542.569	1768.188	53374.004	1913.137
13	XVL.EXE	1.71%	0.256	4011.402	577.732	4615.911	90.187
14	NOTEPAD.EXE	1.00%	0.486	3917.179	1644.862	19528.238	297.881
15	NLNOTES.EXE	1.32%	0.387	3602.648	1306.750	53022.855	1604.908
16	MSOFFICE.EXE	0.88%	0.452	4.067	391.209	76156.078	1339.221
17	EUDORA32.DLL	0.82%	0.694 (13)	596.353	380.499	17111.318	53.847
18	COMCTL32.DLL	0.33%	1.225 (9)	723.196	5067.785	152977.219	2395.858
19	WINHLP32.EXE	0.32%	0.681 (14)	296.368	3365.859	39658.922	1456.344
20	COMDLG32.DLL	0.27%	0.911 (11)	2827.191	5600.863	137626.312	4203.288
21	TELNET.EXE	0.41%	0.148	2668.629	278.863	8942.890	265.787
22	MSACCESS.EXE	0.24%	0.132	6142.746	2278.438	172882.156	564.209
23	SHELL32.DLL	0.27%	2.056 (6)	572.072	3376.698	284729.281	5505.349
24	VBE.DLL	0.21%	0.122	8407.045	793.665	36914.609	306.665
25	WINPROJ.EXE	0.20%	0.035	705.139	857.747	80627.891	192.924
26	SPIRIT.EXE	0.17%	0.012	0.000	98.654	37593.477	79.204
27	MAILNEWS.DLL	0.16%	0.105	8615.952	1727.342	8507.025	670.565
28	ACRORD32.EXE	0.21%	0.059	111.877	1661.644	109964.500	1355.485
29	MPRSERV.DLL	0.14%	0.174	1297.060	729.258	31022.410	424.266
30	RASAPI32.DLL	0.12%	0.335	498.163	1392.521	85277.273	1251.965
31	OTHER-APPS	12.66%	14.781	1925.915	1651.421	93608.320	1629.434

Table 5: The Most Frequently Used Applications ("A#" is the application number, "Application" is the application name, "Time" is the percentage of each application was traced to the total tracing time, "Invoked(r)" is the number of times each application was invoked per hour, "(r)" is the rank of the invoking count, "KeyEvnt/MouseEvnt/FSCall/VMFSCall" are the counts of different events per hour.)

5 User Behavior

We discuss PC user input behavior in this section. First we compare the activity of different user types. Then we study the user input idle pattern. Last, we consider the user input clustering behavior.

5.1 Types of Users

Our traces were collected from two different types of PCs: Desktop PCs and Laptop PCs. Table 6 summarizes the trace event frequencies for these two different types of PC users. The statistics of these two types of users are quite similar. Desktop systems do slightly more file system calls, while laptop users have slightly higher user input frequencies. Although the average main memory size of laptop machines is slightly smaller than the average main memory size of desktop machines, the virtual memory operations on laptop machines are less frequent than desktop machines.

We also categorize the users into three types: engineers, managers, and other. "Other users" include secretaries, assistants, and home PC users. Table 7 compares the trace record frequencies among these three types of users. Managers show the lowest frequency of keyboard input and the lowest application change rate. Secretaries, assistants and home PC users generate the most frequent events in each category. We believe that the difference between keyboard input frequency and mouse input frequency can be interpreted as follows: because a keyboard is a more efficient tool than a mouse for text input while a mouse is more efficient in controlling/information_browsing, the ratio of keyboard activities to mouse activities normally reflects the ratio of the time the PC users spent on inputing text to the time they spent on reading/browsing.

User Type	Desktop User	Laptop User
User#	21	15
KRec/h	1281	1042
MRec/h	838	1495
WRec/h	52	57
FRec/h	56289	52352
VRec/h	1008	712
MM	40MB	48.4MB

Table 6: Desktop User Vs. Laptop User ("User#" is the number of users in this category. "KRec/h", "MRec/h", "WRec/h", "FRec/h", and "VRec/h" are the total numbers of keyboard trace records, mouse trace records, window switch trace records, file system trace records, and virtual memory trace records per tracing hour, respectively. "MM" is the average main memory size.)

As we will also see in our user idle period analysis, the difference between different machine types is small while the differences among different user groups are more obvious. It appears that trace event frequencies or user idle periods are more likely determined by which user group or which workload type, but not by which type of machine being used. The higher rate of virtual memory operations of the group of "Other Users" are due to both higher rate of user activity and the smaller size of main memory in this group of machines.

User Type	Engineers	Manager	Others
User#	23	8	5
KRec/h	1228	738	2127
MRec/h	847	1499	1598
WRec/h	52	39	111
FRec/h	50598	52850	88623
VRec/h	812	623	2188
MM	45.9MB	50MB	32MB

Table 7: Comparison of the Trace Statistics among Different User Types

5.2 User Idle Periods

User idle periods are an interesting topic of study because if the user is not using the system, various components can be powered down. In Section 4, we presented the idle period distribution for the user input, file system and overall system. As may be seen, the overall system is seldom idle for long, whereas the PC users are idle (not generating any input) for a large portion of the time. The cumulative busy period for users is only about 50% with an idle length less than 5 minutes.

Figure 2 illustrates the user input idle period probability density distribution. Note that the spikes at 3-minutes, 4-minutes, 5-minutes and 10-minutes (180, 240, 300, and 600 seconds, respectively) are not normal user behavior. The Windows95 system sometimes posts a few mouse movement messages for resetting the mouse position to an application window. For example, Windows95 will generate a few mouse movement messages to a screen-saver program which is normally configured to be activated after a user idle period of a few minutes. Since these mouse input messages were not from the real users, we can ignore these on-the-minute probability density spikes. We will continue the discussion in next section and give examples of these automatic trace events in next section when we discuss the file system idle period probability density distribution.

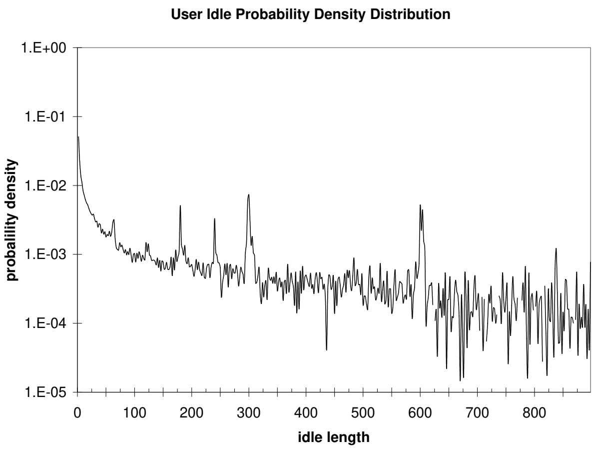


Figure 2: User Idleness Probability Density ("idle length" is measured in seconds.)

Figure 3 compares the user idle period for different user types. The upper plot of Figure 3 shows that there is not much difference between the desktop PC user idle period distribution and that for laptop PC users, while the other plot shows that different user groups have somewhat different idle distributions: the secretary/assistant/homeuser group has the fewest idle periods for long idle length and the engineer user group has fewer idle periods than the manager user group if we only consider the idle periods shorter than one hour. These two plots also agree with the comparison of trace event frequencies shown in Table 6 and Table 7. Please note that our analysis results are based on only the period of tracing time when the machines were powered on, but not when the machines were powered down. Normally, a PC user would turn off his computer at home but not at work when he is not using them. Likewise, in the upper plot of the figure, the variation between the curves on the upper right corner is due to the fact that the laptops are usually turned off when not being used. Appendix III shows user idle behavior for different applications.

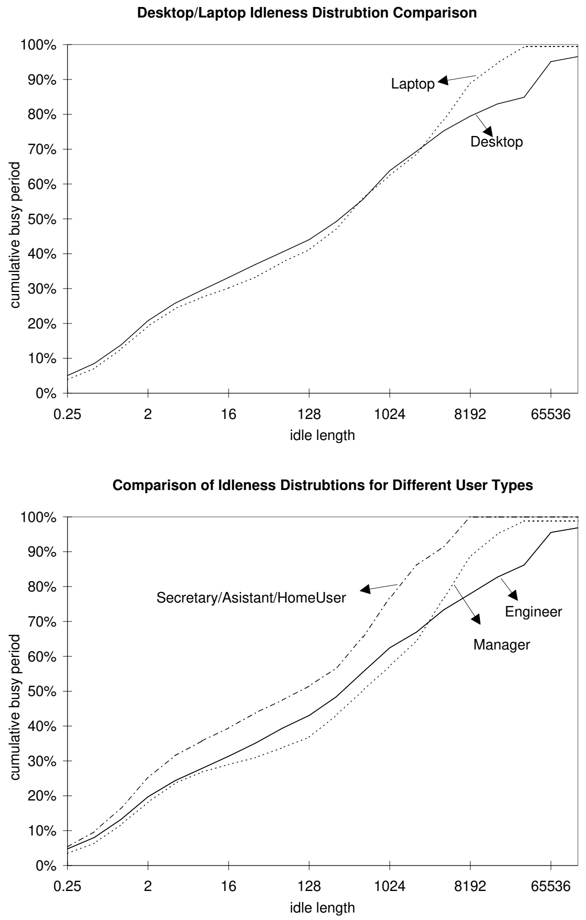


Figure 3: Upper: User Idle/Busy Time Distribution of Different Machine Types; Lower: User Idle/Busy Time Distribution of Different User Types

Figure 4 illustrates the user input idle period as a function of previous user idle period length. Given a time series of user idle period lengths: $t_0, t_1, t_2, ...t_{n-1}, t_n ...$, the previous user idle period length for the nth idle period t_n is t_{n-1}. The user idle period distribution may vary as a function of the length of the previous idle period. Figure 4 plots a set of cumulative busy period distribution curves. Each curve corresponds to the distribution for one previous idle length. For example, the lowest curve is the user idle period distribution when previous idle is within 0.25 second. The next curve is the distribution when previous idle is between 0.25 second and 0.5 second. The figure shows a clear but surprising trend that the longer the previous user idle length, the less chance that the next idle period will be a long one. For example, it is unlikely that there is only one user input event between two long user idle periods.

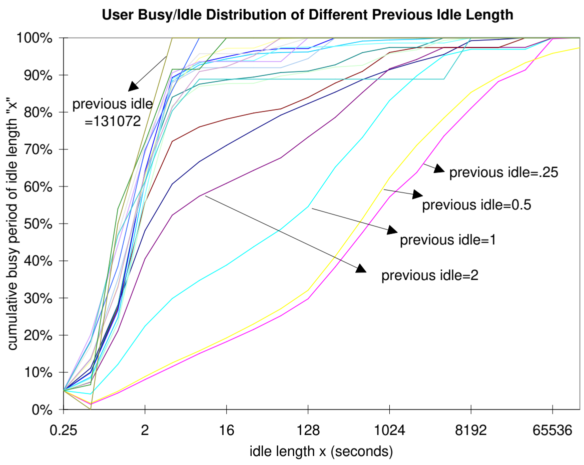


Figure 4: User Idle Time Distribution as function of previous idle period length

5.3 User Input Clustering

In this subsection, we discuss user input clustering patterns. Figure 5 shows the distribution of the number of commands (i.e. application switches) in a user busy period. Since a "busy period" is defined as the period of time with no idle length longer than a certain length of time, we can have different user busy period definitions based on different user idle lengths. Figure 5 plots user commands for busy periods with maximum idle length of 5 seconds, 30 seconds, 1 minute, 2 minutes and 5 minutes in any busy period, respectively. X axis is the number of commands in a busy period. The limit of each bucket on X axis is $[2^n, 2^{n+1} - 1].TheY$ axis is the percentage of tracing time during which there were X number of commands input by the user.

Figure 6 shows the distribution of the number of user input events in a busy period. User in-

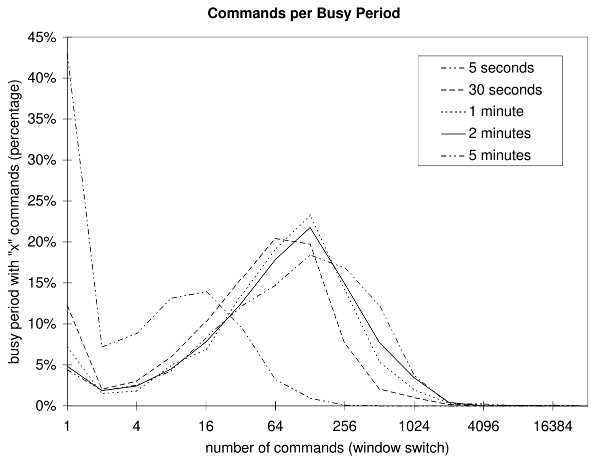


Figure 5: Number of Commands in a Busy Period (for varying busy period definitions)

put events include keyboard input events (key_down and key_up) and mouse input events (movement and button events). As an example, consider the curve for the one minute idle period (i.e. a busy period is one with no idle period over one minute). For 23% of those busy periods, the number of user input events was in the range of 4096 and 8191 (about 2048 to 4095 keystrokes assuming all user inputs are keyboard inputs).

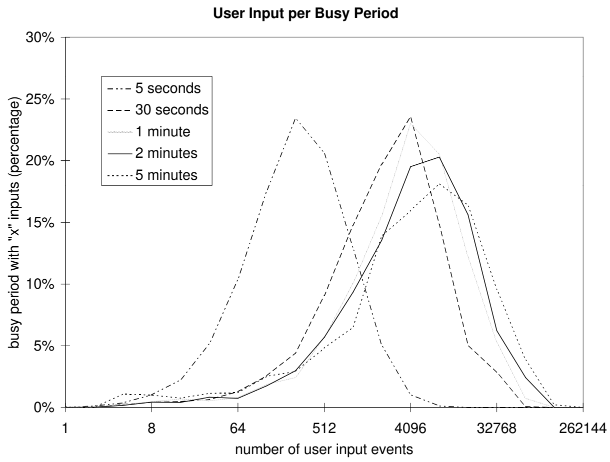


Figure 6: Numer of User Inputs during a Busy Period (for varying busy period definitions)

Table 8 shows the median numbers of commands and user input events for different definitions of the busy period.

Figure 7 shows the transition matrix for the 20 the most frequently used applications. The columns

busy-period	commands	user-inputs
5-seconds	4	256
30-seconds	36	320
1-minute	56	576
2-minutes	60	576
5-minutes	68	704

Table 8: Median Numbers of Commands and User Inputs During a Busy Period

and rows in the matrix correspond to the applications in Table 5. The last matrix column/row represents the category of all other applications. Each entry in the matrix represents the number of times that an application switches to another, per 10,000 application switches. For example, for every 10,000 application switches, there are 71 switches, or 0.71%, on average, from EXPLORER.EXE to NETSCAPE.EXE (matrix[3, 5]).

6 File System Activity

In this section, we first look at frequencies of Windows95 file system calls. Next we analyze the file system idle period patterns. Then we examine different file system function calls. We will also study the READ, WRITE and OPEN file system calls in more detail. Since Windows95 virtual memory paging and swapping operations are identifyable file system operations, we also present data on virtual memory file system calls. Last, we consider file access patterns.

6.1 File System Call Distribution

Table 9 shows the percentage of each of the 20 most frequent file system calls out of the number of total file system calls. The percentages are calculated by the following formula:

$$\frac{\sum_{user=1}^{36} \frac{FunctionCallCount}{AllFSCallCount}}{36} * 100\%$$

where $FunctionCallCount$ is the count of a given file system function call for one user and $AllFSCallCount$ is the count of total file system calls for this user. The various file system calls are further explained in Appendix I.

As may be seen in Table 9, SEEK accounts for 31.06% of the total file system calls. This function call, however, is an advisory file system call, which

	SSAV	MSDO	EXPL	WINW	NETS	SHDO	EUDO	XVIS	MSDE	EXCE	OUTLL	POWE	XVL	NOTE	NLNO	MSOF	EUDO	COMC	WINH	COMD	OTHE	Total
SSAVER	8	19	29	8	7	2	4	7	1	1	2	3	1	5	0	7	11	2	1	0	29	147
MSDOS	10	1310	175	21	33	17	9	36	22	16	3	4	7	15	9	17	1	17	5	9	199	1935
EXPLORER	21	166	730	46	71	45	27	8	25	13	19	13	12	28	4	26	2	70	4	2	453	1785
WINWORD	10	20	46	293	5	21	7	0	0	8	4	2	0	2	1	5	2	1	7	0	42	476
NETSCAPE	7	32	69	4	214	0	2	3	9	1	5	1	0	1	2	13	0	7	0	19	59	448
SHDOCVW	2	15	42	21	1	91	0	2	19	1	1	1	3	3	0	1	0	3	0	9	109	324
EUDORA	12	9	28	9	3	0	4	0	0	1	0	2	0	2	0	2	193	0	0	20	27	312
XVISION	8	32	8	0	3	2	0	21	0	1	0	0	0	7	0	3	0	0	0	0	2	87
MSDEV	1	25	28	0	9	19	0	0	75	0	0	0	0	1	0	0	0	0	0	6	35	199
EXCEL	2	17	13	7	1	1	1	1	0	216	0	0	1	0	1	3	0	0	1	0	8	273
OUTLLIB	1	3	19	3	4	2	0	0	0	0	117	1	0	0	0	3	0	3	1	0	20	177
POWERPNT	3	5	15	3	1	0	2	0	0	0	1	125	0	0	0	0	1	0	0	0	7	163
XVL	1	7	17	0	0	2	0	0	0	0	0	0	6	0	0	0	0	3	1	0	7	44
NOTEPAD	5	16	28	3	1	2	2	7	1	0	0	0	0	10	0	1	1	0	0	5	7	89
NLNOTES	0	9	6	0	1	0	0	0	0	1	0	0	0	0	2	1	0	0	0	1	20	41
MSOFFICE	8	16	45	6	11	0	2	0	0	5	2	1	0	1	0	29	1	2	0	0	11	140
EUDORA32	1	1	1	0	0	0	209	0	0	0	0	0	0	0	0	0	31	0	0	0	2	245
COMCTL32	2	21	20	1	6	3	0	0	0	0	3	0	3	0	0	0	0	12	7	3	126	207
WINHLP32	1	4	5	6	0	0	0	0	0	1	1	0	1	0	0	0	0	7	72	4	12	114
COMDLG32	0	9	2	0	18	9	21	0	6	0	0	0	0	5	2	0	0	3	4	1	66	146
OTHERS	45	204	497	42	60	106	23	1	42	10	20	10	11	8	22	29	2	78	12	65	1337	2624
Total	148	1940	1823	473	449	322	313	86	200	275	178	163	45	88	43	140	245	208	115	144	2578	9976

Figure 7: Application Transition Matrix

N#	TrcEvnt	Function Name	Perc.
1	SEEK	FileSeek	31.06%
2	READ	ReadFile	24.35%
3	FDNXT	FindNextFile	10.22%
4	WRITE	WriteFile	5.14%
5	FDOPN	FindFirstFile	4.10%
6	FNDCL	FindClose	3.84%
7	OPEN	OpenFile	3.84%
8	FATTR	FileAttributes	3.77%
9	CLOSE	CloseFile	3.67%
10	GDSKI	GetDiskInfo	2.62%
11	IOC16	Ioctl16Drive	2.33%
12	FTMES	FileDateTime	1.84%
13	DIR	Dir	0.90%
14	QUERY	QueryResourceInfo	0.59%
15	SEARC	SearchFile	0.44%
16	FLCKS	LockFile	0.40%
17	DSDIO	DirectVolumeAccess	0.32%
18	DELET	DeleteFile	0.16%
19	FLUSH	FlushVolume	0.13%
20	COMMT	CommitFile	0.09%
21	OTHER	Other FS Calls	0.21%

Table 9: Most Used File System Calls ("TrcEvnt" is trace event names; "Perc" is the percentage.)

only manipulates metadata, i.e. the FAT table, and a large portion of the FAT table is normally cached in the main memory. Therefore the SEEK operation does not have a significant impact on the disk IO traffic. Its high frequency is the result of the FAT format file system and Windows95 backward compatibility. An example is a frequently used SEEK operation sequence: first SEEK to the beginning of a file and then SEEK the end of the file. This sequence is used for fetching the whole list of FAT table entries of this file into the main memory, and it is also used for emulating old MS-DOS function calls.

FindNextFile, FindFirstFile and FindClose are directory searching functions. These file system calls are very frequent because in the Windows GUI environment, every file folder open is followed by a set of "Find" file operations. This happens rarely in a UNIX shell environment since normally a UNIX shell user does not always issue an "ls" command following every "cd". FileAttributes, GetDiskInfo, FileDateTime, and Dir are all directory and metadata operations. IoCtl16Drive is a simulated 16-bit direct IO operation, used for backward compatibility. CloseFile flushes out the buffered data to the disks, updates the directories and releases the file handlers. More interesting file system calls are ReadFile, WriteFile and OpenFile. Our further file system study is based on these three types of function calls.

	SEEK	READ	FDNXT	WRITE	FDOPN	FNDCL	OPEN	FATTR	CLOSE	GDSKI	IOC16	FTMES	DIR	QUERY	SEARC	FLCKS	DSDIO	DELET	FLUSH	COMMT	OTHER	Total
SEEK	1321	1581	0	140	2	0	7	4	42	1	1	5	0	0	0	1	0	0	0	0	0	3105
READ	1354	700	1	25	19	0	34	40	219	11	4	4	13	0	1	5	2	0	0	0	0	2432
FDNXT	1	0	876	0	5	79	47	11	0	0	2	0	0	0	0	0	0	0	0	0	0	1021
WRITE	123	20	0	336	1	0	3	1	17	1	4	4	0	0	0	0	0	0	0	4	0	514
FDOPN	2	0	79	0	7	304	5	4	0	1	3	0	0	0	0	0	0	3	0	0	2	410
FNDCL	16	1	5	0	187	1	36	72	0	1	46	0	0	0	2	0	0	12	0	0	3	382
OPEN	76	104	0	5	3	0	21	5	10	1	135	19	1	0	0	3	0	0	0	0	1	384
FATTR	12	2	3	1	120	0	118	91	2	2	3	0	23	0	1	0	0	0	0	0	0	376
CLOSE	54	3	55	2	34	0	78	83	31	3	4	3	8	0	3	1	0	0	1	0	0	363
GDSKI	12	1	0	0	1	0	6	2	1	236	0	0	0	0	2	0	0	0	0	0	0	261
IOC16	0	0	0	0	13	0	1	0	0	0	24	133	0	56	1	0	0	0	3	0	1	232
FTMES	102	20	0	5	0	0	0	4	36	0	0	15	2	0	0	0	0	0	0	0	0	184
DIR	17	1	0	0	2	0	16	7	0	4	0	0	40	0	1	0	5	0	0	0	0	93
QUERY	0	0	0	0	4	0	4	48	0	0	0	0	0	0	0	0	0	0	0	0	2	56
SEARC	1	0	0	0	3	0	1	2	1	1	1	0	2	0	32	0	0	0	0	0	0	44
FLCKS	7	1	0	0	0	0	0	0	2	0	0	0	0	0	0	29	0	0	0	0	0	39
DSDIO	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	25	0	3	0	0	32
DELET	3	0	1	0	4	0	4	1	1	0	0	0	0	0	0	0	0	0	0	0	0	14
FLUSH	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	5	0	5	0	0	13
COMMT	1	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	4	8
OTHER	0	0	1	0	2	0	1	1	1	0	2	0	0	0	0	0	0	0	0	4	2	14
Total	3104	2434	1021	514	407	384	383	376	366	262	233	183	89	56	43	39	37	15	12	8	15	9981

Figure 8: File System Call Transition Matrix

Figure 8 shows the transition matrix for the 20 most frequently used file system calls. The column/row numbers in the matrix: 1, 2, 3, ... , 20 match file system function numbers "N#" in Table 9. The matrix column/row number 21 represents all other file system calls. Each entry in the matrix represents the number of times that the file system call in the column directly follows the call in the row, per 10,000 file system calls. For example, in every 10,000 file system calls, there are about 1581, or 15.81%, SEEKs which are directly followed by READs.

6.2 File System Idle Periods

As seen earlier in Figure 1, unlike user inputs, PC file systems are seldom idle for long periods of time. As shown in the figure, the cumulative file system busy time is about 82% of the total tracing time for an idle length of 512 seconds or less; i.e. during only 18% of the tracing time was the PC file system idle longer than 512 seconds. Please note this statement does not mean that a PC disk drive seldom idles for long period of time. The Windows95 file system caches file system data in the main memory, and thus many operations do not go to the physical disk. This Windows95 file system caching, however, is beyond of the scope of this paper. (See also Table 27 in Appendix III which shows the measured file system idle behavior with logarithmic idle lengths.)

Figure 9 shows the file system idle period probability density distribution. We can observe that there are probability density spikes at idle lengths of 1-minute, 1.5-minutes, 2-minutes, and 5-minutes(60, 90, 120, and 300 seconds, respectively). These on-the-minute spikes, which also show up in the user input idle probability density distribution, are caused by the automatic features of many applications. Examples of these automatic actions, which happen periodically, are SCREEN-SAVER's automatic-startup, WINWORD's automatic-saving and a dynamic HTML down-loaded by the NETSCAPE browser. These periodic events are a major reason that PC file systems usually do not idle for long periods. See Appendix III for file system idle period data for different applications.

6.3 Read/Write Bandwidth

Here we consider the file system READs/ WRITEs and the number of bytes transferred for such a file system call. Table 10 gives the number of bytes per hour transferred by (logical) reads and writes. We see that bytes transferred due to virtual memory paging and swapping accounts for a small part (14.7%) of total bytes transferred. This percentage is smaller than 34.9% reported in [Bake91], and 15%-21% reported in [Beck91]. The difference

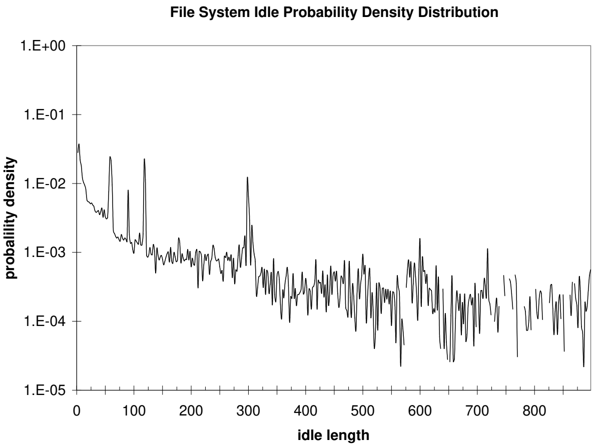


Figure 9: File System Idle Period Probability Density("idle length" is measured in seconds.)

can be explained by two factors: most of the machines we traced have fairly large main memories in comparison to the systems considered in previous studies[Bake91] and [Beck91]; and a Windows95 user normally runs only one application program at a time.

Function	Bytes per hour	Percent
FS Read	37347 K	73.61%
FS Write	5948 K	11.72%
Paging Read	2479 K	4.89%
Paging Write	2874 K	5.66%
Swapping Read	1730 K	3.41%
Swapping Write	361 K	0.71%

Table 10: File System IO Traffic ("FS Read" is regular file system READ, and "FS Write" is regular file system WRITE.)

The file system idle period distribution, discussed in the previous subsection, suggests that the file system IO traffic should be very bursty. Table 11 gives the maximum total IO data transfer to appear in any period of one hour, one minute or 10 seconds, in any of our 36 traces. Also shown in each case is the rate per second. These figures can be compared with Table 10 to see how bursty the file system IO traffic is.

The following two figures, Figure 10 and Figure 11, show the distributions of the number of of file system IO bytes transferred in a period of one hour or one minute. The X axis is the number of

Max TP	READ	WRITE	Total IO
Per Hour (bytes/s)	1424672K (395.7K)	353005K (98.1K)	1480949K (411.4K)
Per Min (bytes/s)	141528K (2358.8K)	94627K (1577.1K)	145466K (2424.4K)
Per 10Sec (bytes/s)	92935K (9293.5K)	36803K (3680.3K)	92935K (9293.5K)

Table 11: File System Maximum IO Traffic Throughput ("Max TP" is the maximum throughput.)

bytes transferred, and the limit of each bucket on X axis is $[2^n, 2^{n+1}-1]$ where n is bucket number. The Y axis is the percentage of tracing time for the different file system IO traffic rates shown on the X axis. For example, the file system total (READ + WRITE) IO bytes transferred per hour were between 8388608 (2^{23}) and 16777215 ($2^{24}-1$) bytes for 17.3% of the total tracing time.

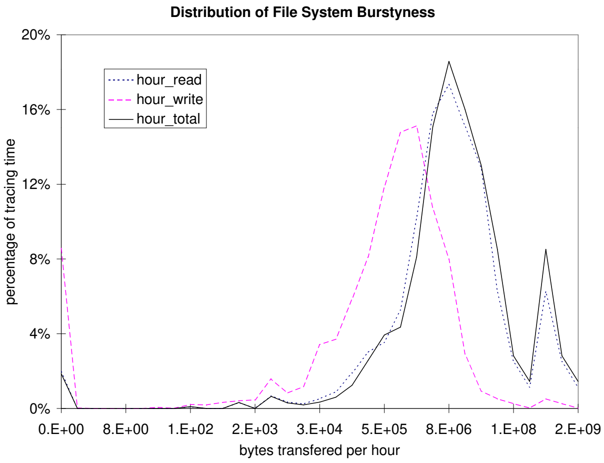


Figure 10: Distribution of File System IO Burstyness (per hour)

The following two figures, Figure 12 and Figure 13, show the distributions of the total number of bytes transferred and number of read/write operations as a function of the block size. The limit of each bucket is $[2^n, 2^{n+1}-1]$ where n is bucket number. The block size is the number of bytes transferred per file system call of the regular file system READs and WRITEs. For example, as shown in Figure 12, 16K bytes were transferred per hour as part of the blocks with size of 4096 to 8191 bytes, with 3900 file READs falling into this range of block sizes.

As can be seen from these figures, most block

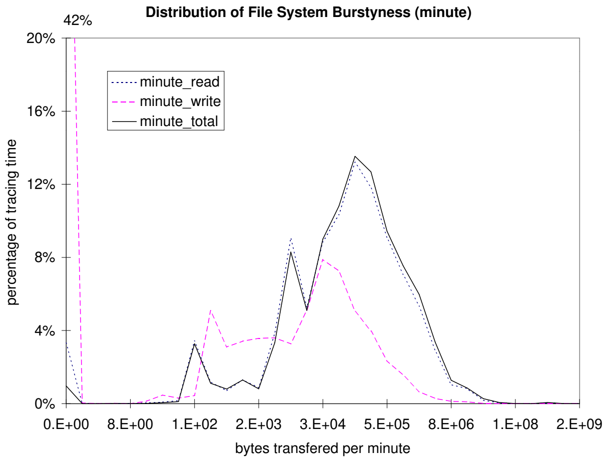


Figure 11: Distribution of File System IO Burstyness (per minute)

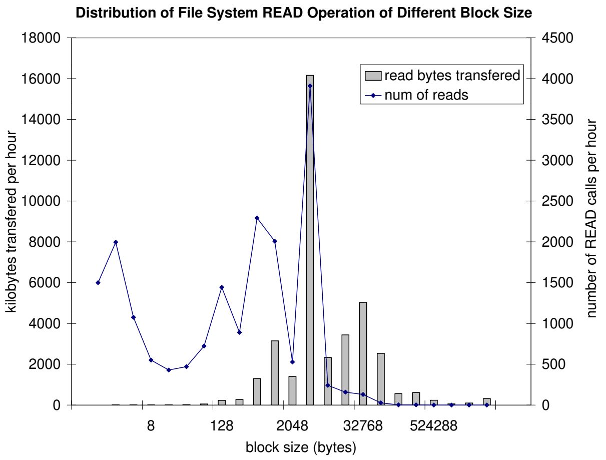


Figure 12: File System READs for Different Block Sizes

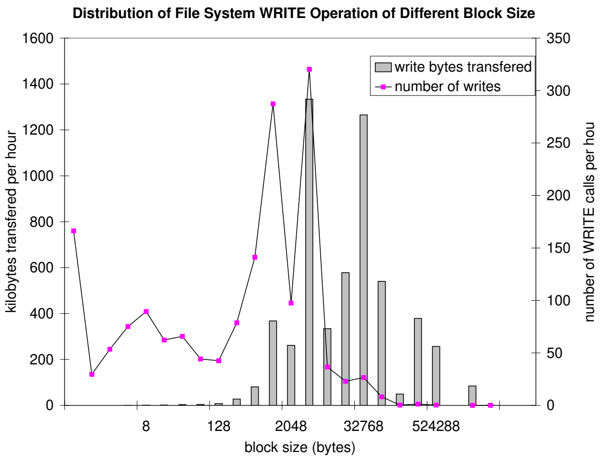


Figure 13: File System WRITEs for Different Block Sizes

sizes are intermediate, 4KB is the most popular size. Since the Windows95 virtual memory page size and the FAT-32 cluster size are both 4K bytes, software designers also tend to use 4KB as the read or write buffer size. This distribution is different from the one in previous study [Cost85] which shows the most frequently used READ block size was 512 bytes, and more than 48% WRITE block sizes were smaller than 1024 bytes. This change is not just Windows95 specific. It is also a natural result of the growth in main memory size, and the increase in the ratio of CPU to I/O speed. Additional discussion and data, specific to the 10 most frequently used applications, appears in Appendix III.

6.4 File Access Patterns

In this subsection, we discuss file access patterns by analyzing file system OPEN operations – 137687 in total in our traces, with 104223 unique files. We also study the distribution of file sizes. and compute the ratio of random IOs to sequential IOs.

Num Acc	Num Files	Num Acc	Num Files
1/8	62920	32	254
1/4	13544	64	129
1/2	11067	128	101
1	7956	256	58
2	5145	512	26
4	1918	1024	11
8	731	2048	2
16	358	4096+	1

Table 12: File Access Distribution ("Num Acc" is the number of open access to one file per 10 hours. "Num Files" is the number of different such files. Note that there are some files which were accessed fewer than once per 10 hours in average.)

Table 12 lists the file access frequency distribution. In the table, we show how frequently a file is accessed by "OPEN" file system calls in a 10-hour period. "Num of accesses" in the table represents the number of accesses to one file in 10 hours. "Num of files" represents the number of different such files that have been accessed the given number of times in a bucket. The limit of each bucket (for integer buckets only) is $[2^n, 2^{n+1}-1]$ where n is bucket number. For example, there are about 358 files among 104223 unique files that have been accessed in an average of 16 to 31 times in every 10 hours. It is interesting to note that on average, 95487 files, or

91.6% of the total PC files, have been accessed only once or fewer during 10 hours. Only 0.6% or 582 files were opened more than 32 times during the same time period. This distribution is derived from the average distribution over 36 trace sets.

File Name	AccessTimes	FileSize
AVCONSOL.INI	45634	8576
10000.DAT	1262	64000
FINDFAST.EXE	1217	130859
WIN.INI	1123	37568
FRONTPG.INI	1102	465
50000.DAT	913	0
MAIN.IND	616	65280
MAIN.IDX	614	3107731
CUSTOM.INI	594	46
ISETUP.INI	593	17571
SYSTEM.INI	509	22480
DESKTOP.INI	479	65024
FONTS.MFM	387	144055
CONTROL.INI	352	6479
PCN.CFG	350	1277
COMMAND.COM	294	116802

Table 13: 16 Most Frequently Accessed Files ("AccessTimes" is the number of accesses per 10 hours. The list excludes Windows shortcut link files which are included in Table 12.)

Table 13 shows the most frequently accessed files. Most of these files are initialization parameter files. Although such files as AVCONSOL.INI, FRONTPG.INI, MAIN.IND, 10000.DAT, and 50000.DAT have a very high access frequency, they are user or application specific files, and show up only in a limited number of user trace sets. Conversely, some Windows component files, such as SYSTEM.INI, WIN.INI, CONTROL.INI, DESKTOP.INI, FINDFAST.EXE and COMMAND.COM show up in most of our user trace datasets.

Since our file system traces do not provide actual file sizes, we use the largest offset of any byte transferred in any I/O to that file as an estimate of the "File Size". For example, an application OPENs a file, then SEEKs to the offset address of 10000'th byte from the beginning of that file, then READs 500 bytes, then CLOSEs it. We say for this round operations, the maximum offset of this file accessed is 10000+500=10500. If this file were opened 5 times, and the largest offset among all maximum offsets was 10500, we use 10500 as the estimate of the size of this file. Table 14 lists a few examples indicating the accuracy of this estimation method. In Table 14, the actual file sizes were gathered from Machine Number 21 of Table 1, and the estimated file sizes were derived from the traces collected from this machine using the above method. We list samples of four different files types in the table, 1) read-only executable files, 2) read-only data files, 3) read-write data files, and 4) temporary files.

file type	file name	e_size	a_size
executable	WINZIP32.EXE	675328	736768
	COMDLG32.DLL	66560	92672
readonly	RMNET.HLP	302343	302343
	COLOR.GMA	0	1050
readwrite	COOKIES.TXT	2056	2080
	SYSTEM.INI	2056	2056
temporary	_INZ0433._MP	501312	501312
	CLASS61.MDM	0	688

Table 14: Estimating the File Sizes ("e_size" is the estimated file size. "a_size" is the actual file size.)

Figure 14 compares the actual file sizes and the estimated file sizes for 388 files shown in the trace of Machine Number 21. Both X axis and Y axes are logarithmic. The figure shows that the majority of file sizes from our estimation are accurate or very close and the actual file size is normally greater than or equal to than the estimated file size. Table 15 illustrates the error distribution of the estimated file sizes. Note that the data presented in this table and figure is very pessimistic. The traces were collected 18 months before the file sizes were specifically collected, and thus many of the errors are due to the file size having changed, not to its having been estimated incorrectly.

ErrRange	Percent	ErrRange	Percent
$(-\infty,-1)$	1.5%	(0.05,0.1]	13.7%
[-1,-0.1)	3.1%	(0.1,0.5]	12.1%
[-0.1,-0.05)	1.0%	(0.5, 1]	25.5%
[-0.05,0.05]	43.0%		

Table 15: Error Distribution of Estimated File Size. ("ErrRange" is calculated with: (actual_file_size-estimated_file_size)/(actual_file_size); "percent" is the percentage of files whose estimated file sizes fall into the corresponding error range.)

Figure 15 illustrates the file size distribution using the above file size estimation method, considering only non-zero file sizes. We exclude zero size files because our estimation method generates many zero byte file sizes [1], but zero byte files are very rare in reality. Therefore we have excluded zero size files to improve the utility and accuracy of our analysis. The X axis in the figure represents the estimated file size in byes. The limit of each bucket on X axis is $[2^n, 2^{n+1}-1]$ where n is bucket number. The Y axis represents the number of different files that have the estimated file size given on the X axis. For example, there are about 6435 different files with estimated file sizes between 2048 bytes and 4095 bytes.

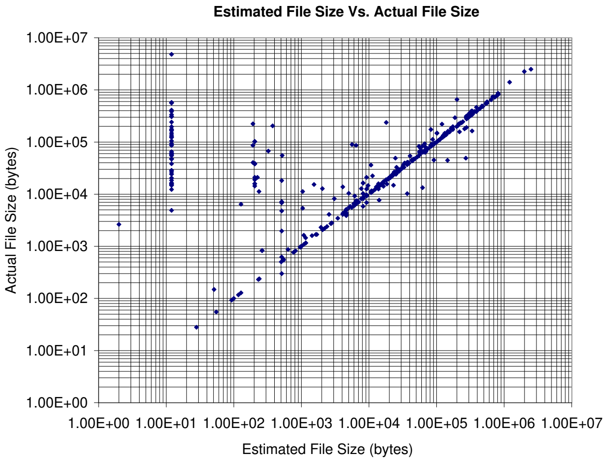


Figure 14: Estimated File Size Vs. Actual File Size

Table 16 lists 10 of the largest files accessed when each of the 10 most frequently used applications were started. Examples of such large files are those DLL files associated with one or multiple applications. Loading an application could be sped up if the system could take advantage of the knowledge of which files need to be accessed before the start of the application.

Table 17 compares the ratios of sequential file system IOs and random file system IOs, in terms of both total bytes transferred and total number of function calls. Please note that we regard the first READ or WRITE call after a file OPEN call as a random IO. As seen in the table, the majority file system IOs are random IOs. The ratio of sequential

[1] There are three SEEK methods: seek from beginning/current_position/end of a file. We cannot compute the logical address in a file with the third SEEK method. Some files were only accessed by the following file system operation sequence: OPEN, SEEK(B), SEEK(E), CLOSE, while some other files were only opened and closed without any READ, WRITE, or SEEK operation against them. We do not know the size of these files. We used 0 for the size of such files.

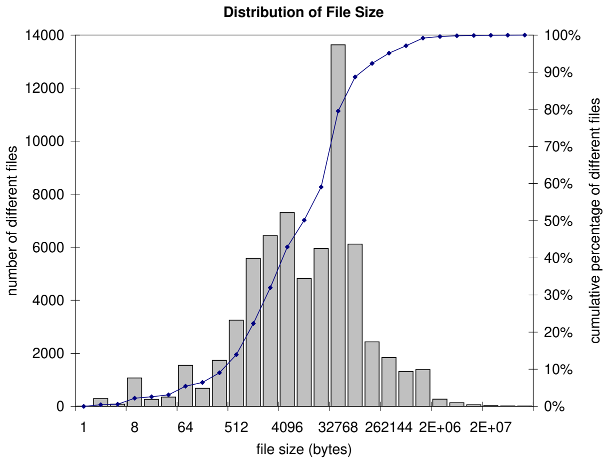


Figure 15: File Size Distribution

IO to random IO for the WRITE file system call is higher than for the READ call, i.e. there are more sequential WRITEs than sequential READs for the same number of WRITE calls and READ calls.

IO Types	Random IOs	Sequential IOs
Bytes Read	87.97%	12.03%
Read Call	93.14%	6.86%
Bytes Written	77.68%	22.32%
Write Calls	75.44%	24.56%

Table 17: File System Random IOs vs. Sequential IOs

7 Miscellaneous Analysis

7.1 Trend Significance Test

In this subsection, we analyze the discrete-time time series of trace events to see if there exists any increasing or decline trend of activities for these time series. We consider each of our 944 continuous tracing period as one time series per file. User input trace events and file system trace events are analyzed separately as two different time series.

In our trend test, we use a statistic described by Lewis and Shedler [Lewi73] to test a time series for being a stationary uncorrelated Poisson process against the alternative of being a Poisson process with monotonic trend. In a discrete-time series trend test, this statistics is as follows (see also

Application	Files Accessed when an Application Started
EXPLORER.EXE	ANAMES.NSF CCALVIN.NSF CONAGENT.EXE OUTBOX.MBX P.ZIP P2.ZIP P3.ZIP P4.ZIP SENTITEMS.MBX VCBKS40.MVB
WINWORD.EXE	ATOK11.DIC CCALVIN.NSF LMSCRIPT.EXE MLANDATA MSGRJP32.LEX MSO97.DLL MSPP32.DLL TTFCACHE WINSPOOL.DRV WWINTL32.DLL
NETSCAPE.EXE	CONAGENT.EXE EXCEL.EXE INBOX INBOX.SNM NETSCAPE.EXE SENT SENT.SNM SPOOLSS.DLL SYSTEM.DAT TRASH
SHDOCVW.DLL	EPG.PCH EXCEL.EXE CONAGENT.EXE OUTBOX.MBX REGSVR32.EXE SYSTEM.DAT TAPIADDR.DLL TLN0LOC.DLL WINWORD.EXE WFM0002.ACV
EUDORA.EXE	ATI.MOD CVS.MBX DESCMAP.PCE EUDORA.EXE IN.MBX POINTLIB.DLL SPOOLSS.DLL SYSTEM.DAT TRASH.MBX WINWORD.EXE
XVISION.EXE	COMDLG32.DLL EXCEL.EXE HOSTS HTML32.CNV MSO97.DLL POINTLIB.DLL SYSTEM.DAT TTFCACHE XV7004.ZIP XVISION.EXE
MSDEV.EXE	BPCAV.PCH BUS_60.AVI CONAGENT.EXE EPG.PCH EXCEL.EXE REGSVR32.EXE SYSTEM.DAT TLN0LOC.DLL VCBKS40.MVB WINWORD.EXE
EXCEL.EXE	ANAMES.NSF COMMAND.WAV CONAGENT.EXE EJLMON21.DLL EXCEL.EXE FUNCRES.XLA MSO97.DLL SORT.WAV SYSTEM.DAT WFM0002.ACV
OUTLLIB.DLL	EXCEL.EXE HTML32.CNV MSO97.DLL OFFLADY.ACT OUTLLIB.DLL MAILBOX.PST MSO97.DLL SPOOLSS.DLL SYSTEM.DAT WINWORD.EXE
POWERPNT.EXE	HOSTS LMSCRIPT.EXE MSO97.DLL OFFICE.CAG POWERPNT.CAG POWERPNT.EXE SYSTEM.DAT USER.DAT WINSPOOL.DRV ~$NORMAL.DOT

Table 16: 10 Largest Files Opened when an Application Starts

[Smit81]):

$$Tr(i) = (S - T/2) / \left(\frac{T}{\sqrt{12 * N(i)}} \right)$$

(where i is the number of this time series; $F(i)$ is its start time, $L(i)$ is its ending time, and j is the current time in this time series; $T = L(i) - F(i) + 1$ is the total duration in seconds of this time series; $N(i)$ is total number of seconds during each of which certain tracing events were logged – either user input events or file system call events depending on the type of time series analyzed; $I(i, j)$ is the indicator function, for Second j in Time-series i, whether certain tracing events were logged – its value is 1 if certain tracing events logged, 0 if not; $S = \sum_{j=F(i)}^{L(i)} \frac{I(i,j)(j-F(i))}{N(i)}$.)

Our trend test shows that 99, or 10.5% of the total 944 valid file system call time series, displayed significant increasing trend, 94, or 10.0% of the total file system call time series displayed significant declining trend, 128, or 13.6% of the total 944 valid user input time series displayed significant increasing trend, and 136, or 14.4% of the total user input time series displayed significant declining trend. Taking into account the fact that the interarrival times for the events in these series are more skewed than exponential, our analysis show no trend effects.

7.2 Serial Correlation Test

We would like to be able to predict or estimate the next idle period, based on the sequence of idle periods thus far. To test the predictability of these two time series, we apply the following first order serial correlation estimator (used by [Smit81]) to the user input idle period series and file system idle period series separately.

The user input input idle time series and the file

$$S_1 = \frac{\frac{1}{n-1}\sum_{i=1}^{n-1}\left(x_i - \frac{1}{n-1}\sum_{i=1}^{n-1} x_i\right)\left(x_{i+1} - \frac{1}{n-1}\sum_{i=1}^{n-1} x_{i+1}\right)}{\left(\frac{1}{n-1}\sum_{i=1}^{n-1}\left(x_i - \frac{1}{n-1}\sum_{i=1}^{n-1} x_i\right)^2 \frac{1}{n-1}\sum_{i=1}^{n-1}\left(x_{i+1} - \frac{1}{n-1}\sum_{i=1}^{n-1} x_{i+1}\right)^2\right)^{1/2}}$$

(where x_i is the idle length, and n is the number of idle periods in one trace file.)

system call idle time series analyzed in the serial correlation test are the same as those used in the trend significance test. The serial correlation test results for both user input idle series and file system idle series shows no significant or usable correlations. Most correlation coefficient values observed were small. Among 944 user input idle series and 944 file system idle series, 908 (96.2%) user input idle series have a correlation coefficient value less than 0.25, 839 (88.9%) user input idle series have such a value less than 0.1, 869 (93.9%) file system idle series have such a value less than 0.25, and 862 (85.0%) file system idle series have such a value less than 0.1. Therefore, we conclude that both the predictive power of user input idle series and that of file system idle series are low.

7.3 Tracing Overhead

In this subsection we discuss the tracing overhead and its potential impact on our results. Because we do not trace processor activities and the user inputs are very sparse, file system tracing dominates our trace. Two types of tracing overheads exist: first, monitoring and generating the trace record; second, dumping the buffered trace records to the hard drives. We do not include processor overhead analysis since processor activities are beyond the scope of our analysis. Our overhead measurement focuses on trace record dumping, i.e. file system call counts contributed by the tracer versus the counts of non-tracing regular file system calls.

Statistical Item	Statistics
Tracer FS Operation Counts	127493
Regular FS Operation Counts	4709118
Tracer FS Overhead	2.67%
Std. Deviation of Overhead	0.326%
90% Confidence Interval	(2.59%, 2.75%)

Table 18: Tracer Operation Overhead

Table 18 shows the tracer overhead and the 90 percent confidence interval (over the 36 samples). We conclude that the trace dumping overhead is relatively insignificant compared to the regular file system activities. Our file system analysis result should not be affected by the trace dumping.

7.4 Limitations of the study

We note the following caveats and limitations in our tracing and analysis:

- A change of window, as discussed in our analysis, does not exactly match an application switch. This causes inaccuracy in our analysis in two ways: 1) An application running in the background may not have an associated window, and our analysis is unable to attribute events to that application. Likewise, an application may be started (e.g. Netscape) and then the user will switch to another window rather than wait. 2) A window switch may occur a few seconds before or after the application switch, so the window switch time used in our analysis is not completely accurate.
- Due to the absence of directory information in the file system traces, we use the largest offset of any byte transferred in any I/O to that file as the file size estimate. In many cases, this underestimates the file size.
- Lack of caching information in our file system analysis. Our disk IO bandwidth analysis is only at the logical level. The ratio of virtual memory physical disk accesses to regular file system physical disk accesses can be different from the logical level ratio because the paging IO cache hit rate is usually much lower than the regular file system IO cache hit rate.

8 Summary and Future work

In this paper, we have presented an analysis of personal computer workloads. Our analysis covers user input behavior and file system activity. Our analysis is based on a large set of PC user and file system traces which were collected from a variety of PC users. The statistics derived from this paper can be used in benchmark development as well as for deriving synthetic workloads for trace driven simulation in different system resource management algorithm studies.

We have provided general descriptive statistics for PC users and file systems. The available data will permit us to do additional analysis of user behavior and file system activities, and establish more complete user and file system statistical models. Trace driven simulation can be applied to the evaluation of various PC system resource management algorithms. We anticipate using this data, and other data currently being collected for WindowsNT systems for studies of disk and I/O optimization, and for power management studies.

9 Acknowledgments

The Authors would like to thank Intel Corp., for its technical support during the tracer development. The authors also thank Intel, Toshiba (Japan), Fujitsu Microsystems (USA), Quantum Corporation, and Sony Research Laboratories (USA) for running the tracer on a number of their machines. We would also like to thank Hugo Patterson of Quantum for comments and suggestions.

Bibliography

[Bake91] Baker, M., Hartman, J., Kupfer, M., Shirriff, K., Ousterhout, J.: Measurements of a Distributed File System *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, Jul. 1991, 1-15.

[Beck91] Becker, J., Park, A.: Analysis of Paging Behavior of UNIX. *Performance Evaluation Review* , Aug. 1991, 36-41.

[Chen96] Chen, J., Endo, Y., Chan, K., Mazieres, D., Dias, A., Seltzer, M., and Smith, M.: The Measured Performance of Personal Computer Operating Systems. *ACM Transactions on Computer Systems.*, Vol. 14, No. 1, Feb. 1996, 3-40.

[Cost85] Da Costa, H.: A File System Tracing Package for Berkeley UNIX. *Technical Report of Computer Science Division (EECS), University of California at Berkeley, No. UCB/CSD-85-235*, Jun. 1985.

[Doug94] Douglis, F., Kaashoek, F., Marsh, B., Cáceres, R., Li, K., Tauber, J. Storage Alternatives for Mobile Computers. *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation*, Monterey, CA, Nov. 1994, 25-37.

[Gold74] Goldberg, R.: Survey of Virtual Machine Research. *IEEE Computer*, Jun. 1974, 34-45.

[Hans85] Hanson, R.: A Characterization of the Use of The UNIX C Shell. *Technical Report of Computer Science Division (EECS), University of California at Berkeley, No. UCB/CSD-86-274*, Dec. 1985.

[Kotz95] Kotz, D., Nieuwejaar, N.: File-System Workload on a Scientific Multiprocessor. *IEEE Parallel and Distributed Technology*, Spring 1995, 51-60.

[Lee98] Lee, D., Crowley, P., Baer, J., Anderson, T., and Bershad, B.: Execution Characteristics of Desktop Applications on Windows NT. *The 25th Annual International Symposium on Computer Architecture.*, Jul. 1998, 27-38.

[Lewi73] Lewis, P. and Shedler, G.: Empirically Derived Micromodels for sequences of Page Exceptions. *IBM Journal of Research and Development*, Vol. 17, Sep. 1973, 86-100.

[Li94] Li, K., Kumpf, R., Horton, P., Anderson, T. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. *Proceedings of the 1994 Winter USENIX Conference*, San Francisco, CA, Jan. 1994, 279-291.

[Lorc97] Lorch, J., Smith, A.: Energy Consumption of Apple Macintosh Computer. *Technical Report of Computer Science Division (EECS), University of California at Berkeley, No. UCB/CSD-97-961*, Jun. 1997.

[Nola74] Nolan, L., Strauss, J. Workload Characterization for Timesharing System Selection. *Software Practice and Experience* , Vol. 4, 1974, 25-39.

[Nort97] Norton, P., Mueller, J.: *Peter Norton's Complete Guide to Windows95, Second Edition*, 0-672-31040-6, Sams Publishing, Indianapolis, IN, 1997.

[Oney96] Oney, W.: *System Programming for Windows95*, Microsoft Press, Redmond, WA, 1996.

[Oust85] Ousterhout, J., Da Costa, H., Harrison, D., Kunze, J., Kupfer, M., Thompson, J.: A Trace-Driven Analysis of the UNIX 4.2 BSD File System. *Proceedings of the 10th Symposium on Operating System Principles*, Orcas Island, WA, Dec. 1985, 15-24.

[Petz96] Petzold, C.: *Programming Windows95*, Microsoft Press, Redmond, WA, 1996.

[Ruem93] Ruemmler, C., Wilkes, J.: UNIX Disk Access Patterns. *Proceedings of the Winter 1993 USENIX Conference*, San Diego, CA, Jan. 1993, 405-420.

[Schu95] Schulman, A.: *Unauthorized Windows95: A Developer's Guide to Exploring the Foundations of Windows*, Chicago, IDG Books, 1995.

[Smit81] Smith, A.: Long Term File Migration: Development and Evaluation of Algorithms. *Communications of the ACM*, Val. 24, No. 8, Aug. 1981, 521-532.

[Smit85] Smith, A.: Disk Cache: Miss Ratio Analysis and Design Considerations. *Proceedings of the 5th annual Symposium on Computer Architecture*, Apr. 1985, 242-248.

[Spas96] Spasojevic, M., Satyanarayanan, M.: An Empirical Study of a Wide-Area Distributed File System. *ACM Transactions on Computer Systems* , Vol 14, No. 2, May 1996, 200-222.

[Zhou85] Zhou, S., Da Costa, H., Smith, A.: A file System Tracing Package for Berkeley UNIX. *Proceedings of USENIX Conference and Exhibition*, Portland, Jun. 1985, 407-419.

[Zhou98] Zhou, M., Smith, A.: Tracing Windows95. Available at URL http://djinn.cs.berkeley.edu/mzhou/tracer.ps.

[Zivk97a] Zivkov, B., Smith, A.: Disk Caching in Large Databases and Timeshared Systems. *Technical Report of Computer Science Division (EECS), University of California at Berkeley, No. UCB/CSD-96-913*, Sep. 1996, Proc. Mascots'97 (Fifth Intl. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems) Conference, Haifa, Israel, January, 1997, pp. 184-195.

[Zivk97b] Zivkov, B., Smith, A.: Disk Cache Design and Performance as Evaluated in Large Timesharing and Database Systems, Proc. CMG (Computer Measurement Group) Conference, December 7-12, 1997, Orlando, FL., pp. 639-658.

Appendix I: Overview of Windows95

In this appendix, we summarize some major characteristics of the Windows95 operating system.

Windows95 is a 32-bit protected-mode operating system designed to run 16-bit and 32-bit application programs on Intel architecture based personal computers. Windows95 uses the VFAT format file system, a version of MS-DOS FAT file system with long filename support. Windows95 provides up to a 4 gigabyte virtual memory. The actual virtual memory size depends on the physical memory and swap space available. Windows95 supports preemptive multitasking of Windows-based and MS-DOS-based applications. Windows95 runs only on PCs based on Intel architecture processors, 80386s or more advanced models. Windows95 does not attempt to provide a secure environment in which program and data can be insulated from another program's inattentive or intentional misbehavior. [Petz96][Oney96][Nort97][Schu95]

A1.1 Windows95 virtual machine

The general concept of virtual machines dates back to early IBM mainframe computers and the work by Robert Goldberg. [Gold74] The virtual machines in the PC world were created when the early versions of Windows needed to support multiple MS-DOS applications and Windows applications running at the same time.[Oney96][Nort97] A virtual machine created by software reacts to application programs the same way a real machine does, which enables the MS-DOS programs to own the keyboard, the mouse, the display screen, the processor, and the user's attention as if they were running on their own dedicated hardware. Specifically, in the kernel of the Windows95 operating system, a Virtual Machine Manager (VMM) manages all virtual machines. The VMM works with Virtual Device Drivers (VxDs) to simulate hardware devices and to provide system services to applications and to each other. There is at least one virtual machine running on a Windows95 system, the system virtual machine, which runs all Windows applications and the Windows95 system itself. One or more MS-DOS virtual machines running MS-DOS applications can co-exist on a Windows95 system.

A1.2 Windows95 memory model

Generally speaking, Windows95 supports three different memory models: the Windows3.1 protected-mode segmented memory model, the WindowsNT flat memory model, and the Virtual-86 model. In the protected-mode segmented memory model, the processor uses a selector (which points to a segment descriptor entry in the memory descriptor table) and an offset pair to reference a memory location. The virtual memory is divided into segments of up to 64KB each. In the flat memory model, there is only one segment which contains all the programs. Virtual memory with a two-level page table paging scheme is used where each 32-bit address is split into three fields: page table directory pointer, page table pointer, and page offset. Each page frame is 4K bytes. In the virtual-86 mode, 20-bit addresses yield only 1MB of address space. A segment/offset pair is used to generate the 20-bit memory address.

A1.3 Windows95 processes and threads

Each Windows application occupies a process that consists of a dedicated address space and one or more threads of execution. Each thread corresponds to a sequence of program steps and the evolving state of processor registers and system objects associated with that sequence. Windows95 uses a priority-based scheme to preemptively multi-task threads.

Windows95 supports three types of applications: Windows 32-bit application programs, Windows 16-bit application programs, and MS-DOS application programs. Both 32-bit and 16-bit Windows application programs run on the system virtual machine while each MS-DOS application programs run on a separate MS-DOS virtual machine. The system virtual machine has one process for each program, and each 32-bit Windows program can consist of more than one thread. The additional virtual machines are for MS-DOS programs, and each contains exactly one process and one thread.

The 32-bit Windows programs adopt the flat memory model, wherein all code and data can be addressed in a single segment covering all of the virtual memory. The 16-bit Windows programs use the Windows3.1 segmented memory model, in which available virtual memory is subdivided into segments of up to 64 KB each. The 16-bit Windows programs load segment selectors into the processor's segment registers to access more than 64 KB of memory. The 32-bit programs participate in preemptive multitasking under the overall control of the scheduling subsystem of the virtual machine manager, while the 16-bit Windows applications must cooperatively multi-task amongst themselves – from this point of view, sometimes Windows95 is not viewed as a preemptive multitasking system. MS-DOS program multitasking depends on the scheduling among different virtual machines.

Most of the time, one or a few windows are associated with one Windows program. Similarly to the UNIX foreground process, a user-input-focused window in Windows is the foreground window to which the user input will be posted.

A1.4 Window95 file system

Windows95 uses an installable file system manager (IFS manager), the highest layer in the file system, to handle all file system calls from Windows 32-bit applications, Windows 16-bit applications and MS-DOS applications. We will discuss IFS in the next subsection in more detail. The IFS manager calls on file system drivers (FSDs) to support different file system formats. The file system formats currently supported by Windows95 include FAT-16 (File Allocation Table with 16 bit entries), FAT-32 (32 bit FAT entry version of FAT, used in the OSR2 (OEM Service Release 2) version of Windows95 and Windows98 and the CD-ROM file system. The FSDs in turn talk to disk drivers which interface with the hardware directly.

A FAT (including FAT-16 and FAT-32) format disk consist of a BOOT sector, a file allocation table, a root directory, and a cluster section. BOOT stores the basic information about the disk and for the use of system boot. The root directory stores the information describing each file entry in the top level directory. The disk cluster section is divided into separated clusters. The notion of a cluster, which is a contiguous collection of disk sectors, was introduced as the allocation unit. Each FAT table entry is used to maintain the status of a disk cluster, and the number of FAT table entries is equal to the number of the clusters on a disk. A FAT table is organized as a linear array containing multiple one-way linked lists. One list corresponds to a file or sub-directory. The FAT entry location of the head of each list is stored in the root directory or a sub-directory. In a FAT file system, sub-directories are stored as regular files.

FAT-16 uses a fixed FAT table size (32KB), 16-bit FAT table entries, and variable cluster sizes. FAT-16 supports up to 2GB per logical hard drive. A hard drive larger than 2GB needs to be partitioned into a few logical hard drives for a FAT-16 format file system. For example, FAT-16 uses 32KB cluster for a 2GB hard drive, 16KB cluster for a 1GB hard drive, ... , 4KB cluster for a 128MB hard drive, etc. Different from FAT-16, FAT-32 uses a variable FAT table size, 32-bit FAT table entries, and a fixed cluster size (4KB). It supports up to a 2TB hard drive. Our target systems all use the FAT-16 format file system for their hard drives.

The FAT file system used in Windows95 file system is called VFAT, virtual FAT – an improved version of the old MS-DOS FAT format file system plus long filename support. Similar to FAT, VFAT also can be classified as VFAT-16 and VFAT-32. The VFAT file system has two file names for each file, a DOS-8.3 format filename (maximum 8 bytes for the file name and maximum 3 bytes for the file name extension) and Windows95 specific long filenames which can be as long as 256 bytes.

A1.5 Windows95 Installable File System and IFS Calls

The file systems of both Windows3.1 and MS-DOS depend on MS-DOS's INT21 code to manage files on disk. Since MS-DOS INT21 is not reentrant, multiple processes cannot simultaneously perform file system calls without proceeding one at a time through this critical section. Windows95 relies on the Installable File System Manager to solve this problem and support asynchronous I/Os. All file system calls of Windows 32-

bit applications, Windows 16-bit applications and MS-DOS applications go to the IFS manager. These file system calls include the accesses to the memory swap file as well. IFS manager calls on FSDs to implement diverse file systems like FAT and the CD-ROM file system. The FSDs talk to disk drivers which interface with the hardware components such as hard drive and floppies directly.

The IFS manager exports a number of virtual device driver level services for use by other parts of the system. These IFS services and Windows95 virtual device driver's dynamic loading, which was designed for plug-and-play, allows third party software and hardware vendors to write their own device drivers as part of Windows95 file system. One of the most important services provided by IFS manager is the IFS_Mgr_InstallFileSystemApiHook service. IFS_Mgr_InstallFileSystemApiHook takes the address of the user VxD hook procedure as an argument, and it returns the address of another hook procedure. A VxD hook procedure is a VxD procedure which will be triggered when the hook-targeting system service is invoked. All the VxD hook procedures should chain the call instead of just processing it to give other potential hooks their chance to examine each request to the targeted system services. Internally, the IFS manager maintains its own list of API hooks so that the users can add and remove the hooks in any order.

There are 31 most commonly used Windows95 installable file system calls generated by IFS manager. These calls are our file system tracing targets. Next we give the names of and an explanation for these installable file system calls.

- *FS_ReadFile* transfers data from the file to a memory buffer. The memory buffer can be filled asynchronously using one or more I/O requests. In a regular FSD implementation, Windows95 VCACHE facilities should be used to maintain a cache of disk records to minimize the physical I/O.
- *FS_WriteFile* transfers data from a memory buffer to the file. A cache of disk-sector-sized buffers containing the data should be maintained and the physical write operations should be performed asynchronously.
- *FS_FileSeek* is an advisory service that allows an FSD to optimize its prefetches of a file. This function is advisory because the read and write functions both supply a file position that overrides anything recorded by the FSD.
- *FS_OpenFile* takes indicated actions to open a file which matches the parsed pathname.
- *FS_CloseFile* flushes any output buffers to disk, deletes internal structures related to the file, and generally cleans up after a series of operations on an open file.
- *FS_CommitFile* flushes buffered data of a file handle to disk.
- *FS_EnumerateHandle* enumerates file handle information.
- *FS_HandleInfo* gets and sets information for a file by the file handle.
- *FS_LockFile* locks or unlocks a byte range in a file by the file handle.
- *FS_FileDateTime* sets or retrieves the time-stamps which are associated with an open file. There are three Windows95 file time-stamps: creation time, last-modified time, and last-accessed time.
- *FS_DeleteFile* deletes the files whose parsed pathname appears in the request pathname.
- *FS_Dir* performs a function on a directory. Directory functions include creating, deleting, checking for the existence of a directory, or converting a directory name between its long-name form and its 8.3 form.
- *FS_DirectDiskIO* is called by IFS manager to handle MS-DOS INT 25h and INT 26h (absolute disk read and write) requests.
- *FS_DirectVolumeAccess* performs direct volume (file system storage resource logical unit) accesses.
- *FS_ConnectNetResource* connects or mounts a network resource.
- *FS_DisconnectResource* is the function to take the actions required when one of the FSD volumes is unloaded or deleted.
- *FS_FileAttributes* gets or sets the attributes of a file.
- *FS_FindChangeNotifyClose* and *FS_FindNextChangeNotify* search for file change notifies on a certain disk drive.
- *FS_FindFirstFile*, *FS_FindNextFile* and *FS_FindClose* go together to implement a normal file search. *FS_FindFirstFile* initiates a file search that can include wildcards, and creates a context handle. *FS_FindNextFile* continues the search with the context handle until no more matches are possible. *FS_FindClose* closes the context handle.
- *FS_FlushVolume* flushes any pending output data to the device.
- *FS_GetDiskInfo* retrieves information about the free space on a disk drive.
- *FS_GetDiskParms* returns the real-mode address of an MS-DOS disk parameter block.
- *FS_Ioctl16Drive* performs an I/O control operation on the volume.
- *FS_QueryResourceInfo* provides basic information about the file system to the IFS manager.
- *FS_RenameFile* renames one or more files. Wildcards in the source name can be specified by the user.

- *FS_SearchFile* is the MS-DOS equivalent of the *FS_FindFirstFile* family of functions.
- *FS_TransactNamedPipe* performs named pipe operations.
- *FS_UNCPipeRequest* performs UNC path based named pipe operations.

A1.6 Windows95 frequently used 30 applications

- *SCREEN-SAVER* Screen saver is a set of automatic programs that start if the computer has been idle for the number of minutes specified by the user.
- *MSDOS-PROMPT* MSDOS-Prompt is a system application for providing MS-DOS application running environment, a MSDOS virtual machine in Windows95 system. Under Windows95, all MS-DOS applications are started in a MSDOS-Prompt Window and running in the MSDOS virtual machine. All MS-DOS application programs are categorized as the MSDOS-Prompt application in our analysis.
- *EXPLORER.EXE* Explorer is the Windows95 desktop user interface application that provides both program management and file management. It offers both a one-pane and a two-pane interfaces for the file management, and a taskbar interface for the program management.
- *WINWORD.EXE* WinWord is the Microsoft word processing application. It is one of the major applications in Microsoft Office groupware applications.
- *NETSCAPE.EXE* Netscape is a well known Internet browsing application from Netscape Corp. Its formal name is Netscape Navigator.
- *SHDOCVW.DLL* SHDOCVW.DLL is a component of Windows95 system. It is known as the system shell document object and control library. It is in the format of DLL (dynamic linked library).
- *EUDORA.EXE* Eudora is the Microsoft Windows POP/SMTP mailer application.
- *XVISION.EXE* XVision is an X server Windows application, which is also known as Hummingbird Exceed from Hummingbird Corp.
- *MSDEV.EXE* MSDev is known as the Microsoft developer studio application. It is the common interface to most Microsoft Windows software development tools: Microsoft Visual C++, Microsoft Fortran PowerStation, Microsoft Visual Test, Microsoft Developer Network, and Microsoft Visual J++.
- *EXCEL.EXE* Excel is the Microsoft spreadsheet application and is part of Microsoft Office groupware.
- *OUTLLIB.DLL* OUTLLIB.DLL is the Microsoft Office OUTLOOK dynamic linked library.
- *POWERPNT.EXE* PowerPNT (powerpoint) is the Microsoft presentation application, one of the Microsoft Office groupware applications.
- *XVL.EXE* XVL is a component of XVision (X server for Windows) application from Hummingbird Corp.
- *NOTEPAD.EXE* Notepad is a small Microsoft text file editor application. It is the default Windows text editor program.
- *NLNOTES.EXE* NLNotes is known as Lotus Notes, a well known office application from Lotus.
- *MSOFFICE.EXE* MSOffice is the Microsoft Office shortcut bar application.
- *EUDORA32.DLL* EUDORA32.DLL is the dynamic linked library part of Microsoft Windows POP/SMTP mailer application.
- *COMCTL32.DLL* COMCTL32.DLL is a component of Windows95 system. It is known as the custom control library. It is in the format of DLL (dynamic linked library).
- *WINHLP32.EXE* WinHLP32 is the Microsoft help application which reads WINHELP format files and provides online helping information. WINHELP is the standard Windows software help file format.
- *COMDLG32.DLL* COMDLG32.DLL is a component of Windows95 system. It is known as the common dialog library which provides dialog features to all the Windows application. It is in the format of DLL (dynamic linked library).
- *TELNET.EXE* Telnet is a Windows application which provides PC users the telnet remote login environment. There are many different versions of this application from different software vendors.
- *MSACCESS.EXE* MSAccess is the Microsoft personal database management application. It is also known as ACCESS, which is a component of the professional version of Microsoft Office software.
- *SHELL32.DLL* SHELL32.DLL is a component of Windows95 system. It is known as Windows shell common dynamic linked library.
- *VBE.DLL* VBE.DLL is a component of Microsoft common shared libraries. It is known as the VESA BIOS Extensions dynamic linked library.
- *WINPROJ.EXE* WinProj is a Microsoft project management software.
- *SPIRIT.EXE* Spirit is a Windows application tool specific only to a few of our tracing target PC systems.
- *MAILNEWS.DLL* MAILNEWS.DLL is a system mail/news dynamic linked library installed on the Windows95 systems with Microsoft Internet Explorer (IE3/IE4).

- *ACRORD32.EXE* AcroRD32 is known as 32 bit version of Acrobat Reader from Adobe Corp. It is an application for reading PDF format documents.
- *MPRSERV.DLL* MPRSERV.DLL is a component of Windows95 system. It is known as the Multinet Router program library. It is in the format of DLL (dynamic linked library).
- *RASAPI32.DLL* RASAPI32.DLL is a component of Windows95 system. It is known as Dial-Up Network Dynamic Linked Library or Remote Access 32-bit API Dynamic Linked Library.

Appendix II: WMonitor trace record data files

Trace record data files record the following data: USER_ID, StartTime, StopTime, and the trace records. StartTime and StopTime contain the Windows95 internal millisecond counter when the WMonitor starts and stops instrumenting the system activities. The Windows95 internal millisecond counter is the elapsed (integer) time in milliseconds since the Windows95 system's most recent start. Each trace record includes four data fields: time stamp, trace type, function name, and information detail. We further discuss the trace record structure in next subsection. All numbers are hexadecimal numbers except the number in USER_ID record in a trace record data file. One trace record spans exactly one text line with the return and line-feed characters, *0D* and *0A* in ASCII code, as the line separator. We can determine the calendar date and time for StartTime, StopTime and each trace record based on the time-stamp and the readings of the StartTime record and the StartDate record in the corresponding WMonitor system profile log file.

The following file is a WMonitor trace record data file example:

```
USER_ID: 761
StartTime: 9E9C4F
Time  Type  Funct   Details
130   3     OPEN    C: [223] \DAT\WMONITOR.INI
0     3     WRITE   C: [223] 93
0     3     CLOSE   C: [223]
0     3     SEEK    C: [2A8] 4B400:B
0     3     READ    C: [2A8] 200
A     2     [e54] C:\WMONITOR\BIN\WMONITOR.EXE
0     3     READ    C: [271] 1000 MM
0     3     FATTR   C: \WINDOWS\SYSTEM\MFC40LOC.DLL
0     3     FDOPN   C: \WMONITOR\BIN\*.*
DB    0     K_DN    11
96    0     K_UP    91
0     3     FLCKS   C: [26B]
0     3     RENAM   C: \DAT\DATA.ZIP \RECYCLED\DCO.ZIP
0     3     DIR     C: QLGD \WMONITOR\BIN\MSGHK.DLL
0     3     DELET   C: \RECYCLED\DESKTOP.INI
0     1     START_MV
9E    1     STOP_MV
... ...
Stop_Time: 1D41D3C
```

With detailed information and time stamps for every trace event available, WMonitor trace record data files can be used in comprehensive workload analysis and tracing driven simulations. Except for the calendar date and time information, WMonitor system activity profiling information log files can be reproduced from the trace record data files.

A2.1 WMonitor trace record structure

Each trace record in a WMonitor trace record data file includes up to four data fields: time stamp, trace type, function name, and detail information. The trace record data fields are separated by the character of ASCII code 09. Each trace record contains up to 539 bytes. When a trace record reaches its maximum length, two full Windows95 long file pathnames, each of 256 bytes, are included.

The time stamp records the time when a traced event triggers a WMonitor procedure. The incremental time stamp is used to reduce the record size. The absolute time stamp can be derived by accumulating the incremental time stamps and then adding the StartTime. The granularity of time stamp is one millisecond. The upper limit of this time stamp is 0xFFFFFFFF.

Trace type can have one of the following four values:

- 0 – keyboard input event
- 1 – mouse or other pointing device input event
- 2 – user-input-focused window switch event
- 3 – file system call event

Table 19 also lists all trace record types in a WMonitor trace record data file. Different types of trace records interpret the function name field and detail information field differently, which we will discuss in the following two sub-sections.

A2.2 User activity trace record

There are three types of user activity trace records: keyboard input record, mouse input record and user-input-focused window switch record.

- keyboard input record: For a keyboard input event, the function name is either "K_DN" (pressing a key) or "K_UP" (releasing a key). The 1 byte (7 valid bits) key scan code is stored in the detail information field. The 8th bit of this byte indicates the status of the key being accessed: 1 – up and 0 – down.

  For example, if the input is a capital ASCII "*K*", four keyboard trace events are recorded: 0x2A (scan code of "left_shift") K_DN, 0x25 (scan code of "*k*") K_DN, 0xA5 (scan code of "*k*" + 0x80) K_UP, and 0xAA (scan code of "left_shift" + 0x80) K_UP, where "left_shift" K_DN and "left_shift" K_UP are not generated if Caps_Lock is in function.

Trace type	Explanation	Data field
USER_ID	user id	user-identification
StartTime	start time	trace-starting-time*
Stop_Time	stop time	trace-stopping-time*
0	keyboard input	keyboard-event key-scancode
1	mouse input	mouse-event
2	window switch	[window-handle] application-name
3	file system call	see Table 20

Table 19: WMonitor Trace Records (* is the Windows internal millisecond counter readings when tracing starts/stops.)

- mouse input record: For a mouse input event, the function name can be one of the following mouse events: L_DWN (pressing the left mouse button), L_UP (releasing left mouse button), L_CLK (double clicking the left mouse button), M_DWN (pressing middle mouse button), M_UP (releasing middle mouse button), M_CLK (double clicking middle mouse button), R_DWN (pressing right mouse button), R_UP (releasing right mouse button), R_CLK(double clicking right mouse button), START_MV(starting moving the mouse), and STOP_MV(stopping moving the mouse). The detail information field is empty for the mouse input event case.
- user-input-focused window switch record: For a window switch event, the window handle and the application software full pathname are stored in the function name field and detail information field, respectively.

A2.3 File system call trace record

Table 20 gives a list of file system function call names, the corresponding detail information fields, and installable file system call names. We discussed the installable file system calls previously.

File system call trace records include both regular file access call records and memory swap file access call records. Memory swap file access call records are mapped memory reads, mapped memory writes, memory paging reads, or memory paging writes. Memory swap file access records distinguish themselves from regular file access records by the last two bytes in the detail information field: either "MM" (Mapped Memory) or "PG" (memory PaGing).

Appendix III: Miscellaneous Data

Table 21, Table 22, Table 23, Table 24, and Table 25 show the same statistics, which we have seen in Table 5, of 30 of the most frequently run applications among desktop users, laptop users, manager type users, engineer type users and other users, respectively.

Table 26 shows the measured user input idle behavior with logarithmic scaled idle lengths.

The 4 small plots in Figure 16 are used to show the different user idle period behaviors among the most frequently used applications. In each plot, we give both the user input idle time distribution of 4 specific applications in dotted lines and the user input idle time distribution of all programs with a solid line. PC users behave differently when running different applications. The plots show that PC users are more active, and tend to not idle for a long time with an application whose dotted line is above the solid line. Examples of such software tools are WINWORD, NETSCAPE, EXCEL, and NOTEPAD etc. The plots also show that PC users' idle behavior is roughly average when using system software such as EXPLORER, EUDORA, and MSOFFICE etc. Obviously, PC users are inactive when SCREEN-SAVER is running.

Table 27 shows the measured file system idle behavior with logarithmic scaled idle lengths.

We use 4 small plots in Figure 17 to show the measured file system idle behaviors of the most frequently used applications. In each plot, we give both the file system idle time distribution for four specific applications in dotted lines and the file system idle time distribution for all programs with a solid line. It can be seen how the file system idle pattern varies with different running applications. PC file systems idle less time for such applications as WINWORD, NETSCAPE, EUDORA, EXCEL, and POWERPNT than for other applications. The file system idle behaviors of most other applications are about the average.

Figure 18 and Figure 19 repeat the same analysis we did in Figure 12 with additional data for 10 of the most frequently used applications. Applications such as EUDORA.EXE, XVISION, and SCREEN-SAVER have significantly fewer calls and transfer fewer bytes than other applications. Applications with a large number of READs, such as NETSCAPE.EXE and MSDOS-PROMPT, have fewer large sized READs than other applications such as MSDEV.EXE.

Similarly, Figure 20 and Figure 21 show the distri-

function name	detail information field	IFS call name
READ	diskdrive fhandle[1] bytes vm_opt[2]	*FS_ReadFile*
WRITE	diskdrive fhandle bytes vm_opt	*FS_WriteFile*
FDNXT	diskdrive handle[3]	*FS_FindNextFile*
FCNNT	diskdrive	*FS_FindNextChangeNotify*
SEEK	diskdrive fhandle bytes position[4]	*FS_FileSeek*
CLOSE	diskdrive fhandle	*FS_CloseFile*
COMMT	diskdrive fhandle	*FS_CommitFile*
FLCKS	diskdrive fhandle	*FS_LockFile*
FTMES	diskdrive fhandle	*FS_FileDateTime*
PIPRQ	diskdrive	*FS_TransactNamedPipe*
HDINF	diskdrive fhandle	*FS_HandleInfo*
ENMHD	diskdrive	*FS_EnumerateHandle*
FNDCL	diskdrive handle[3]	*FS_FindClose*
FCNCL	diskdrive	*FS_FindChangeNotifyClose*
CNNCT	diskdrive	*FS_ConnectNetResource*
DELET	diskdrive filename	*FS_DeleteFile*
DIR	diskdrive method[5] filename	*FS_Dir*
FATTR	diskdrive filename	*FS_FileAttributes*
FLUSH	diskdrive	*FS_FlushVolume*
GDSKI	diskdrive	*FS_GetDiskInfo*
OPEN	diskdrive fhandle filename	*FS_OpenFile*
RENAM	diskdrive filename1 filename2	*FS_RenameFile*
SEARC	diskdrive filename	*FS_SearchFile*
QUERY	diskdrive	*FS_QueryResourceInfo*
DISCN	diskdrive	*FS_DisconnectResource*
UNCPR	diskdrive	*FS_UNCPipeRequest*
IOC16	diskdrive	*FS_Ioctl16Drive*
GDSPR	diskdrive	*FS_GetDiskParms*
FDOPN	diskdrive filename	*FS_FindFirstFile*
DSDIO	diskdrive	*FS_DirectVolumeAccess*

Table 20: File System Call Records (fhandle[1] is the file handle in the format of "[hex-number]". vm_opt[2] is the virtual memory operation indicator which can be null (i.e. " "), or one of "PG" or "MM". handle[3] of FDNXT and FNDCL is the file searching context handle. position[4] can be one of "begin", "end", or "current". method[5] can be one of "mkdir", "rmdir", "chechdir", "query8.3dir", or "querylongdir".)

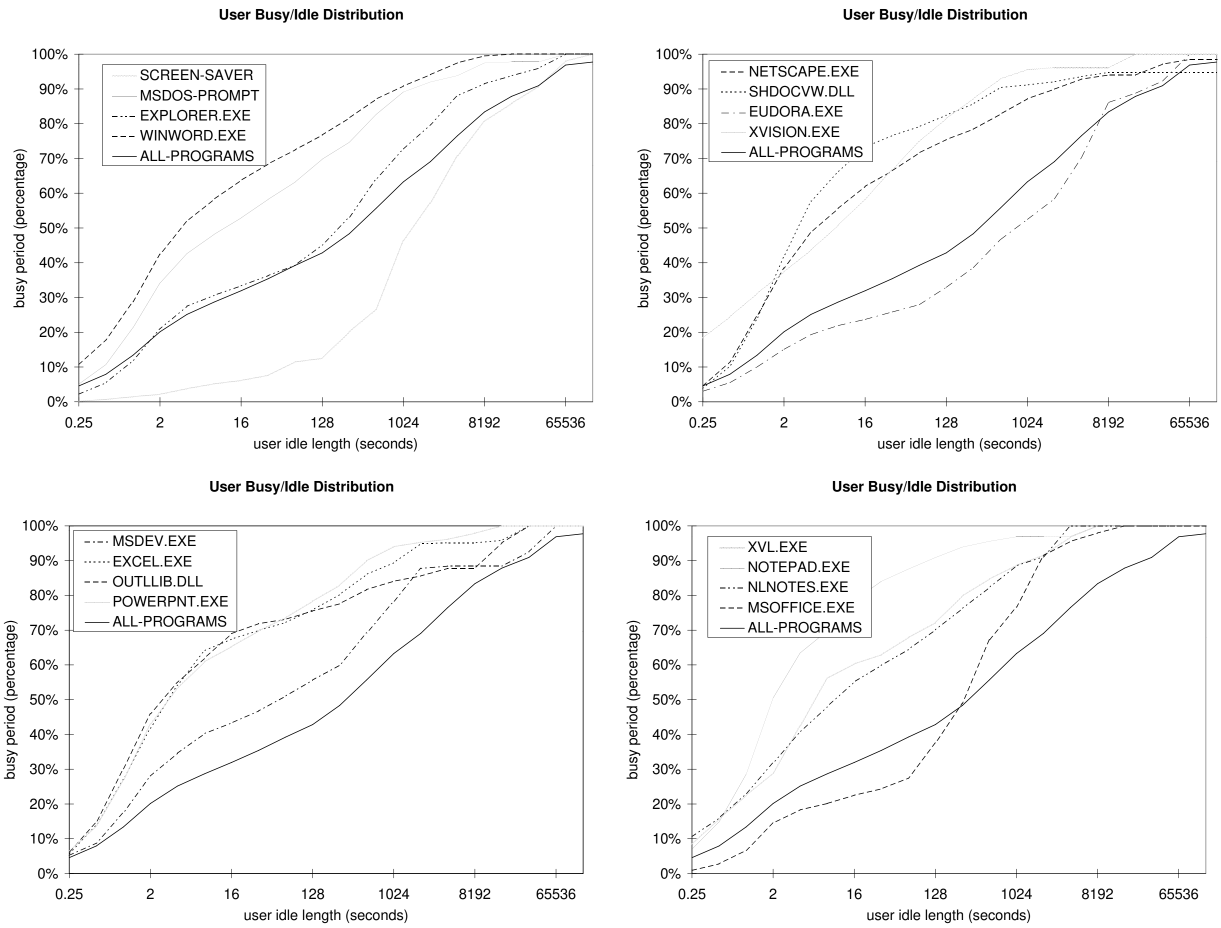


Figure 16: User Idle/Busy Time Distribution of Different Applications

A#	Application	Time(r)	Invoked(r')	KeyEvnt	MsEvnt	FSCall	VMFSCall
1	SCREEN-SAVER	20.19%(1)	0.553	0.769	105.090	87967.398	119.467
2	MSDOS-PROMPT	11.19%(2)	17.992(1)	2772.651	2146.047	44422.184	1531.045
3	EXPLORER.EXE	9.81%(3)	10.803(2)	458.039	1113.267	59752.309	2100.458
4	WINWORD.EXE	7.29%(4)	2.868(3)	3765.487	1847.313	108861.945	4249.324
5	NETSCAPE.EXE	6.87%(5)	2.242(6)	861.056	1266.012	94241.625	1726.872
6	SHDOCVW.DLL	5.37%(8)	2.738(5)	429.762	2635.267	101092.172	3459.110
7	EUDORA.EXE	0.36%	0.040	21.120	195.367	20102.994	250.122
8	XVISION.EXE	6.62%(6)	0.896(10)	7694.425	226.582	57960.312	319.893
9	MSDEV.EXE	5.89%(7)	2.865(4)	3131.524	1251.985	69845.750	3572.496
10	EXCEL.EXE	1.99%	1.469(8)	2552.007	3019.337	45318.383	766.171
11	OUTLLIB.DLL	2.98%(9)	0.986(9)	2705.327	654.595	151393.312	552.313
12	POWERPNT.EXE	0.18%	0.237	1722.181	2641.297	51856.781	3273.526
13	XVL.EXE	2.92%(10)	0.439	4011.402	577.732	4615.911	90.187
14	NOTEPAD.EXE	0.95%	0.601	3927.248	1867.932	30435.764	467.469
15	NLNOTES.EXE	0.02%	0.022	6894.016	1418.306	83230.367	827.870
16	MSOFFICE.EXE	0.77%	0.395	0.536	328.966	110597.719	1685.465
17	EUDORA32.DLL	0.03%	0.017	165.146	269.645	29028.352	323.761
18	COMCTL32.DLL	0.39%	1.776(7)	969.748	5905.655	197024.812	2868.450
19	WINHLP32.EXE	0.29%	0.781	443.907	3697.854	53124.121	1607.634
20	COMDLG32.DLL	0.31%	0.913	2564.405	5447.927	108565.922	3953.821
21	TELNET.EXE	0.66%	0.214	2694.514	214.075	9322.622	278.365
22	MSACCESS.EXE	0.42%	0.226	6142.745	2278.439	172882.188	564.209
23	SHELL32.DLL	0.24%	2.157	554.103	3478.052	214584.375	5394.480
24	VBE.DLL	0.36%	0.206	8428.405	787.312	36320.055	244.545
25	WINPROJ.EXE	0.00%	0.003	0.000	6243.163	442208.062	7972.039
26	SPIRIT.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
27	MAILNEWS.DLL	0.28%	0.179	8615.952	1727.342	8507.025	670.565
28	ACRORD32.EXE	0.25%	0.084	117.509	631.685	122621.852	995.427
29	MPRSERV.DLL	0.12%	0.102	938.017	515.811	34634.395	463.589
30	RASAPI32.DLL	0.05%	0.149	95.432	1493.428	80613.844	2069.950
31	OTHER-APPS	12.87%	16.926	1586.767	1622.998	97288.414	1809.894

Table 21: The Most Frequently Used Applications for Desktop Users ("A#" is the application number; "Application" is the application name, "Time(r)" is the percentage of time each application was traced to the total tracing time, "(r)" is the rank of tracing time. "Invoked(r')" is the number of times each application was invoked per hour, "(r')" is the rank of the invoking count, "KeyEvnt/MsEvnt/FSCall/VMFSCall" are the counts of different events per hour. "n/a" represents "not available".)

butions of bytes transferred and number of calls between for WRITE block sizes for the 10 most frequently used applications.

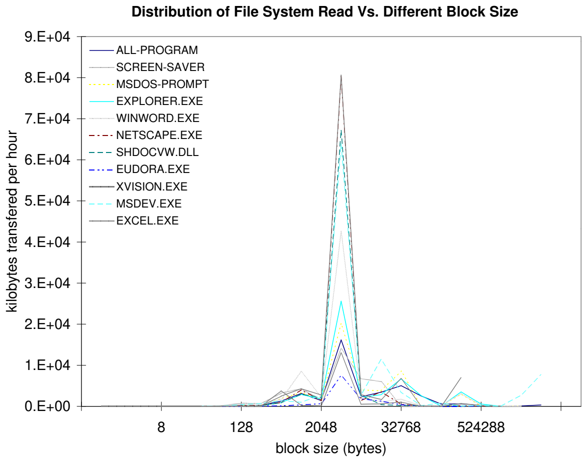


Figure 18: Bytes Transfered of READ Operations vs. Different Block Sizes of Different Applications

A#	Application	Time(r)	Invoked(r')	KeyEvnt	MsEvnt	FSCall	VMFSCall
1	SCREEN-SAVER	25.18%(1)	0.739	0.184	30.044	74983.461	59.302
2	MSDOS-PROMPT	12.88%(2)	24.718(1)	1698.112	2715.631	30858.846	678.259
3	EXPLORER.EXE	7.26%(4)	8.314(2)	175.156	1301.458	51159.625	1535.693
4	WINWORD.EXE	7.25%(5)	2.581(5)	3773.160	1802.811	61485.605	2097.828
5	NETSCAPE.EXE	3.38%(7)	1.455(9)	988.197	3001.398	109353.859	1865.646
6	SHDOCVW.DLL	1.43%	3.279(3)	1546.347	2975.285	159004.953	8189.200
7	EUDORA.EXE	11.30%(3)	2.044(6)	769.065	663.863	26354.199	71.171
8	XVISION.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
9	MSDEV.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
10	EXCEL.EXE	3.27%(8)	3.017(4)	2179.602	3440.749	25738.100	2490.340
11	OUTLLIB.DLL	0.96%	1.178(10)	3793.645	2856.020	137692.156	892.109
12	POWERPNT.EXE	4.35%(6)	1.139	1532.013	1716.874	53463.180	1833.185
13	XVL.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
14	NOTEPAD.EXE	1.07%	0.326	3904.690	1368.169	5998.755	87.528
15	NLNOTES.EXE	3.14%(9)	0.898	3579.865	1305.978	52813.758	1610.286
16	MSOFFICE.EXE	1.03%	0.532	7.770	456.484	40036.484	976.108
17	EUDORA32.DLL	1.92%(10)	1.642(8)	605.422	382.831	16860.672	48.170
18	COMCTL32.DLL	0.25%	0.453	175.362	3206.048	55103.969	1345.763
19	WINHLP32.EXE	0.37%	0.541	134.061	3000.632	24845.887	1289.911
20	COMDLG32.DLL	0.22%	0.908	3342.541	5900.784	194616.797	4692.518
21	TELNET.EXE	0.05%	0.056	2150.984	1574.466	1349.159	14.258
22	MSACCESS.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
23	SHELL32.DLL	0.30%	1.914(7)	592.402	3262.018	364096.406	5630.792
24	VBE.DLL	0.00%	0.003	0.000	3293.961	270903.344	24754.617
25	WINPROJ.EXE	0.49%	0.079	706.423	847.938	79969.289	178.754
26	SPIRIT.EXE	0.41%	0.028	0.000	98.654	37593.477	79.204
27	MAILNEWS.DLL	0.00%	0.000	n/a	n/a	n/a	n/a
28	ACRORD32.EXE	0.15%	0.025	98.938	4028.051	80883.180	2182.745
29	MPRSERV.DLL	0.18%	0.275	1621.910	922.378	27754.391	388.687
30	RASAPI32.DLL	0.21%	0.596	633.282	1358.666	86841.891	977.528
31	OTHER-APPS	12.36%	11.777	2420.533	1692.875	88241.203	1366.250

Table 22: The Most Frequently Used Applications for Laptop Users

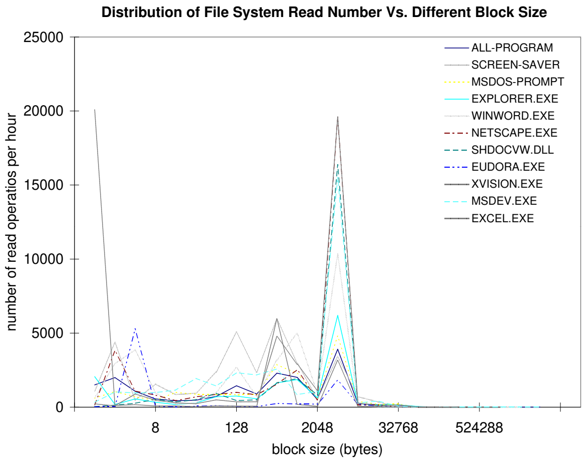


Figure 19: File System READ Operations Number vs. Different Block Sizes of Different Applications

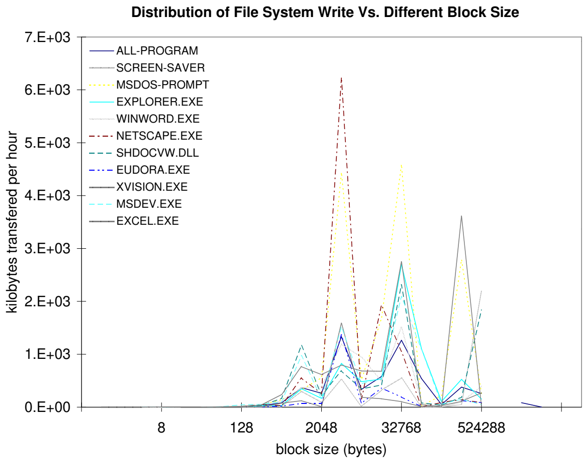


Figure 20: Bytes Transfered of WRITE Operations vs. Different Block Sizes of Different Applications

A#	Application	Time(r)	Invoked(r')	KeyEvnt	MsEvnt	FSCall	VMFSCall
1	SCREEN-SAVER	17.09%(1)	0.789	2.270	15.898	43490.238	75.252
2	MSDOS-PROMPT	7.65%(5)	6.797(3)	603.063	4441.306	30333.857	787.215
3	EXPLORER.EXE	8.36%(4)	7.309(1)	191.558	1180.832	54394.719	1638.405
4	WINWORD.EXE	10.22%(3)	2.192(5)	2994.873	1123.016	85271.336	939.740
5	NETSCAPE.EXE	4.81%(6)	1.712(7)	319.880	2225.503	134431.297	1098.030
6	SHDOCVW.DLL	4.03%(8)	6.901(2)	1116.682	3088.711	129143.391	6147.991
7	EUDORA.EXE	14.77%(2)	2.256(4)	639.260	615.575	26418.029	73.226
8	XVISION.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
9	MSDEV.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
10	EXCEL.EXE	1.05%	0.623	1755.755	2675.299	57397.484	2354.195
11	OUTLLIB.DLL	1.46%	0.570	276.403	835.789	125518.109	106.047
12	POWERPNT.EXE	4.59%(7)	0.866(10)	1002.484	1566.688	43581.023	436.418
13	XVL.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
14	NOTEPAD.EXE	1.17%	0.665	1496.969	759.597	4144.574	246.772
15	NLNOTES.EXE	2.19%(10)	0.308	2624.637	1093.262	46710.875	3211.693
16	MSOFFICE.EXE	2.30%(9)	0.816	6.454	321.730	91846.250	539.777
17	EUDORA32.DLL	1.92%	1.814(6)	244.221	386.780	19264.875	25.867
18	COMCTL32.DLL	0.27%	0.666	316.344	4026.753	84427.758	2196.889
19	WINHLP32.EXE	0.09%	0.188	321.772	3106.099	53475.039	864.920
20	COMDLG32.DLL	0.26%	1.088(9)	4661.635	5319.023	254170.938	5323.927
21	TELNET.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
22	MSACCESS.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
23	SHELL32.DLL	0.26%	1.226(8)	541.496	3674.675	455546.688	9042.322
24	VBE.DLL	0.00%	0.000	n/a	n/a	n/a	n/a
25	WINPROJ.EXE	0.92%	0.147	705.617	854.433	80335.086	182.145
26	SPIRIT.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
27	MAILNEWS.DLL	0.70%	0.404	8739.584	1670.039	6257.145	575.281
28	ACRORD32.EXE	0.81%	0.129	34.720	1492.520	111912.125	762.068
29	MPRSERV.DLL	0.08%	0.211	1311.225	1648.725	44862.289	865.152
30	RASAPI32.DLL	0.14%	0.494	1344.398	2227.797	56293.805	2776.335
31	OTHER-APPS	13.79%	10.620	2673.142	1422.441	109011.352	1991.548

Table 23: The Most Frequently Used Applications for Manager Type Users

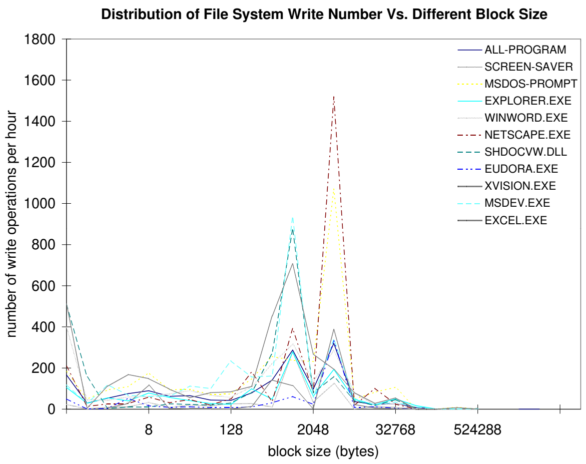


Figure 21: File System WRITE Operations Numbers vs. Different Block Sizes of Different Applications

A#	Application	Time(r)	Invoked(r')	KeyEvnt	MsEvnt	FSCall	VMFSCall
1	SCREEN-SAVER	25.96%(1)	0.643	0.139	89.116	62497.566	91.956
2	MSDOS-PROMPT	8.87%(3)	15.680(1)	2190.405	2034.186	46164.996	1568.427
3	EXPLORER.EXE	9.24%(2)	10.305(2)	380.564	1080.156	54092.098	1507.099
4	WINWORD.EXE	5.06%(7)	2.228(5)	4780.083	2135.123	87498.164	2297.597
5	NETSCAPE.EXE	5.98%(5)	2.037(6)	933.823	1391.715	78027.180	2049.089
6	SHDOCVW.DLL	3.52%(8)	1.691(8)	476.971	2420.674	102444.695	1925.847
7	EUDORA.EXE	2.56%	0.584	932.815	700.222	25419.334	90.148
8	XVISION.EXE	6.05%(4)	0.818	7694.425	226.582	57960.312	319.893
9	MSDEV.EXE	5.38%(6)	2.616(4)	3131.524	1251.985	69845.750	3572.496
10	EXCEL.EXE	2.91%(9)	2.692(3)	2694.053	3480.675	29598.010	1862.773
11	OUTLLIB.DLL	2.22%	0.708	3252.901	628.178	157123.641	646.350
12	POWERPNT.EXE	1.02%	0.417	1479.633	1956.064	28789.926	2522.706
13	XVL.EXE	2.67%(10)	0.401	4011.402	577.732	4615.910	90.187
14	NOTEPAD.EXE	1.05%	0.484	4640.805	1982.406	22913.561	299.226
15	NLNOTES.EXE	1.30%	0.499	4174.174	1431.508	56711.430	665.941
16	MSOFFICE.EXE	0.40%	0.294	0.890	491.598	55217.277	2135.177
17	EUDORA32.DLL	0.61%	0.455	981.155	373.635	14757.955	84.422
18	COMCTL32.DLL	0.40%	1.608(9)	848.314	5291.124	171492.312	2507.759
19	WINHLP32.EXE	0.45%	0.920(10)	277.457	3308.933	36323.160	1300.672
20	COMDLG32.DLL	0.30%	0.904	2532.241	5705.828	108048.227	3894.737
21	TELNET.EXE	0.31%	0.171	4666.352	470.625	11891.114	516.739
22	MSACCESS.EXE	0.38%	0.193	6215.635	2230.134	173006.000	473.225
23	SHELL32.DLL	0.29%	2.027(7)	625.306	2942.151	252742.172	4233.720
24	VBE.DLL	0.32%	0.165	8543.806	741.658	36372.191	165.521
25	WINPROJ.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
26	SPIRIT.EXE	0.27%	0.018	0.000	98.654	37593.477	79.204
27	MAILNEWS.DLL	0.01%	0.023	4751.519	3518.516	78833.273	3648.938
28	ACRORD32.EXE	0.04%	0.048	624.129	2784.463	97034.109	5295.204
29	MPRSERV.DLL	0.18%	0.177	1383.181	594.297	20218.197	217.350
30	RASAPI32.DLL	0.12%	0.283	244.227	947.914	98993.305	334.433
31	OTHER-APPS	11.85%	16.651	1455.495	1960.222	103894.641	1741.169

Table 24: The Most Frequently Used Applications for Engineer Type Users

A#	Application	Time(r)	Invoked(r')	KeyEvnt	MsEvnt	FSCall	VMFSCall
1	SCREEN-SAVER	13.58%(2)	0.320	0.028	7.743	329207.812	115.780
2	MSDOS-PROMPT	32.63%(1)	66.718(1)	3041.361	2099.848	31462.283	753.221
3	EXPLORER.EXE	7.11%(4)	11.219(2)	555.639	1760.458	77331.359	4785.117
4	WINWORD.EXE	12.73%(3)	6.030(3)	2911.969	2174.929	97259.594	8393.284
5	NETSCAPE.EXE	3.79%(6)	1.670(8)	1772.296	3046.427	170730.250	1036.458
6	SHDOCVW.DLL	4.20%(5)	2.520(6)	334.792	3114.330	112037.203	10083.119
7	EUDORA.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
8	XVISION.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
9	MSDEV.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
10	EXCEL.EXE	3.11%(7)	1.840(7)	1198.493	2549.112	44701.109	623.354
11	OUTLLIB.DLL	2.86%(8)	3.509(5)	3834.659	2826.006	138223.938	924.655
12	POWERPNT.EXE	1.80%(9)	1.108(9)	3913.581	2103.139	157331.094	6363.308
13	XVL.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
14	NOTEPAD.EXE	0.54%	0.211	5812.422	1691.610	42350.117	461.572
15	NLNOTES.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
16	MSOFFICE.EXE	0.80%	0.594	0.304	482.811	51554.863	3205.181
17	EUDORA32.DLL	0.00%	0.000	n/a	n/a	n/a	n/a
18	COMCTL32.DLL	0.11%	0.355	165.966	5348.107	104309.852	1229.585
19	WINHLP32.EXE	0.12%	0.366	587.671	4628.859	80322.172	4773.610
20	COMDLG32.DLL	0.16%	0.658(10)	665.555	5418.014	93872.234	3990.986
21	TELNET.EXE	1.50%(10)	0.281	786.732	98.220	6165.603	29.385
22	MSACCESS.EXE	0.02%	0.063	0.000	6349.296	162446.172	8231.838
23	SHELL32.DLL	0.16%	3.515(4)	194.749	6326.953	102685.852	6992.895
24	VBE.DLL	0.03%	0.115	364.109	3852.163	68813.672	8607.277
25	WINPROJ.EXE	0.00%	0.015	0.000	5739.186	511935.344	16069.720
26	SPIRIT.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
27	MAILNEWS.DLL	0.00%	0.000	n/a	n/a	n/a	n/a
28	ACRORD32.EXE	0.00%	0.000	n/a	n/a	n/a	n/a
29	MPRSERV.DLL	0.11%	0.102	615.354	688.585	98153.148	1502.833
30	RASAPI32.DLL	0.10%	0.322	42.922	2042.578	72421.891	3039.882
31	OTHER-APPS	14.54%	12.833	2555.819	841.064	31665.572	660.975

Table 25: The Most Frequently Used Applications for Other Type Users

idle length (sec')	1/4	1/2	1	2	4	8	16	32	64	128
cumu. busy peri.	4.6	7.9	13.4	20.1	25.1	28.7	31.9	35.4	39.3	42.8
idle length (sec')	256	512	1024	2048	4096	8192	16384	32768	66k	131k
cumu. busy peri.	48.3	55.7	63.3	69.1	76.5	83.4	87.8	90.9	96.9	97.7

Table 26: User Idle/Busy Time Distribution("idle length" is user input idle length in seconds. "cumu. busy peri." is the percentage of cumulative busy period.)

idle length (sec')	1/4	1/2	1	2	4	8	16	32	64	128
cumu. busy peri.	1.6	16.9	22.1	31.1	33.9	39.9	44.8	49.3	60.0	68.4
idle length (sec')	256	512	1024	2048	4096	8192	16384	32768	66k	131k
cumu. busy peri.	73.9	81.7	86.7	89.8	91.8	95.5	95.7	96.2	96.6	96.6

Table 27: File System Idle/Busy Time Distribution("idle length" is file system idle length in seconds. "cumu. busy peri." is the percentage of cumulative busy periods.)

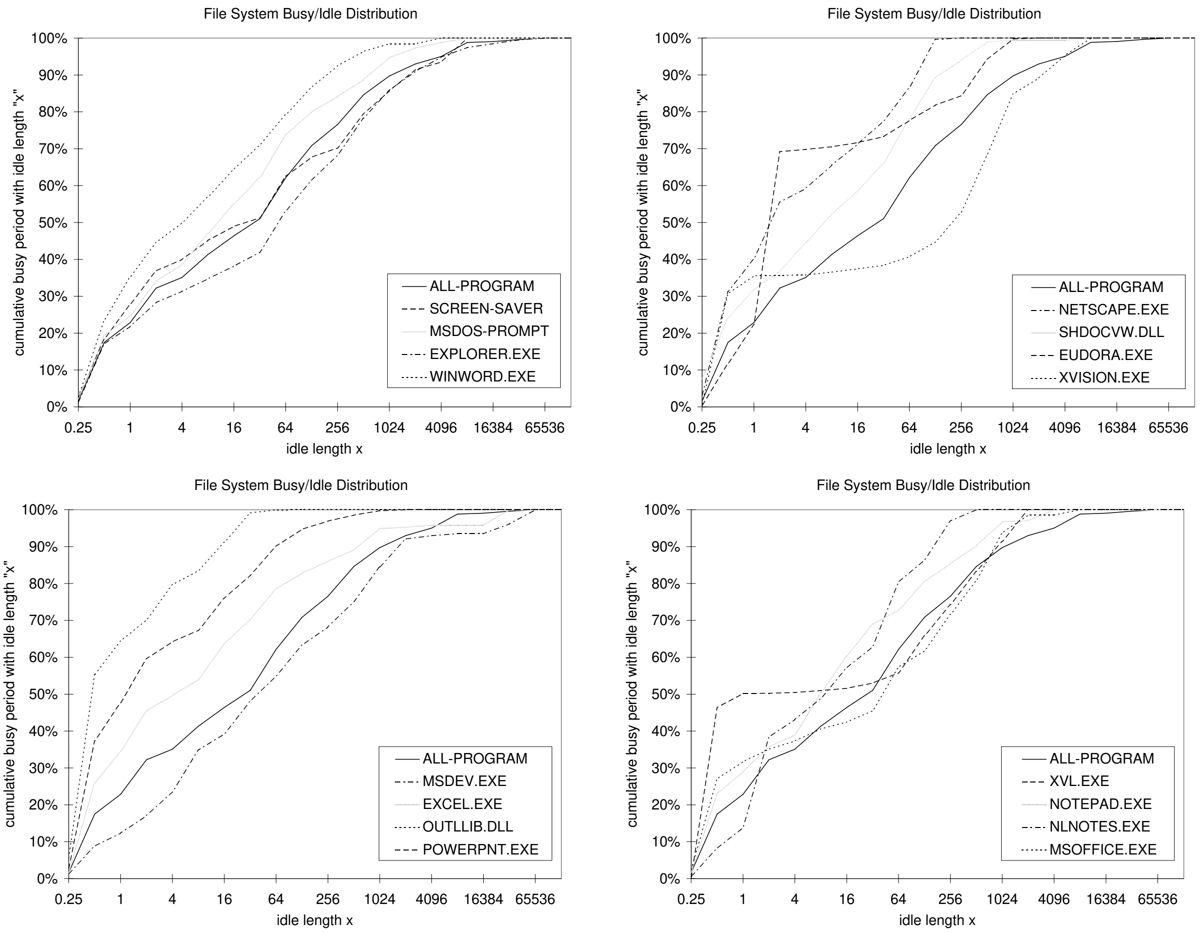


Figure 17: File System Idle/Busy Time Distribution of Different Applications