Appendix A4

Appendix for Chapter 4

This chapter appendix provides additional graphs and tables for the workload characterization in Chapter 4. Section A4.1 displays supplementary graphs for the general analysis in Section 4.2. Section A4.2 presents several graphs for the 12:00 pm to 8:00 pm interval to determine which time interval is chosen for the model. A detailed analysis of the current operation of the CDF system is provided in Section A4.3. Section A4.4 discusses the resource usage of frequently used commands.

A4.1 General Analysis

The graphs in this section were generated from the data that were produced by the UNIX `top` command. Each graph shows data for a four day period on one host. A representative subset of the hosts in the CDF system was chosen for display in this section.

A4.1.1 UNIX Load Averages

Figure A4.1 displays four day UNIX load averages at 12 minute intervals on 12 different hosts. Figure A4.1 (a) reveals that the UNIX load average on marvin follows a cyclic, daily behaviour. During the early hours of the morning the load reaches a daily high. The load reaches a daily low before noon, and then rises again in the afternoon. It drops at about 6:00 pm each evening, and then gradually begins to climb again. The early morning system activity is because of the `earlymorn` script that was discussed in the Chapter 3. The increase in load after 12:00 pm was likely caused by user requests that were propagated to the file server during this busy period of user activity in the system.

The location of peaks and the shape of eddie's load graph in Figure A4.1 (b) [1] are similar to the load graph for marvin. Periods of correspondingly heavy load on these two servers was especially noticeable during the day when user activity dominated system activity on eddie. This correspondence is plausible because whenever a user on eddie requested a file that was not found on eddie's local disk, the request was transmitted to marvin's Prestoserve or disk(s).

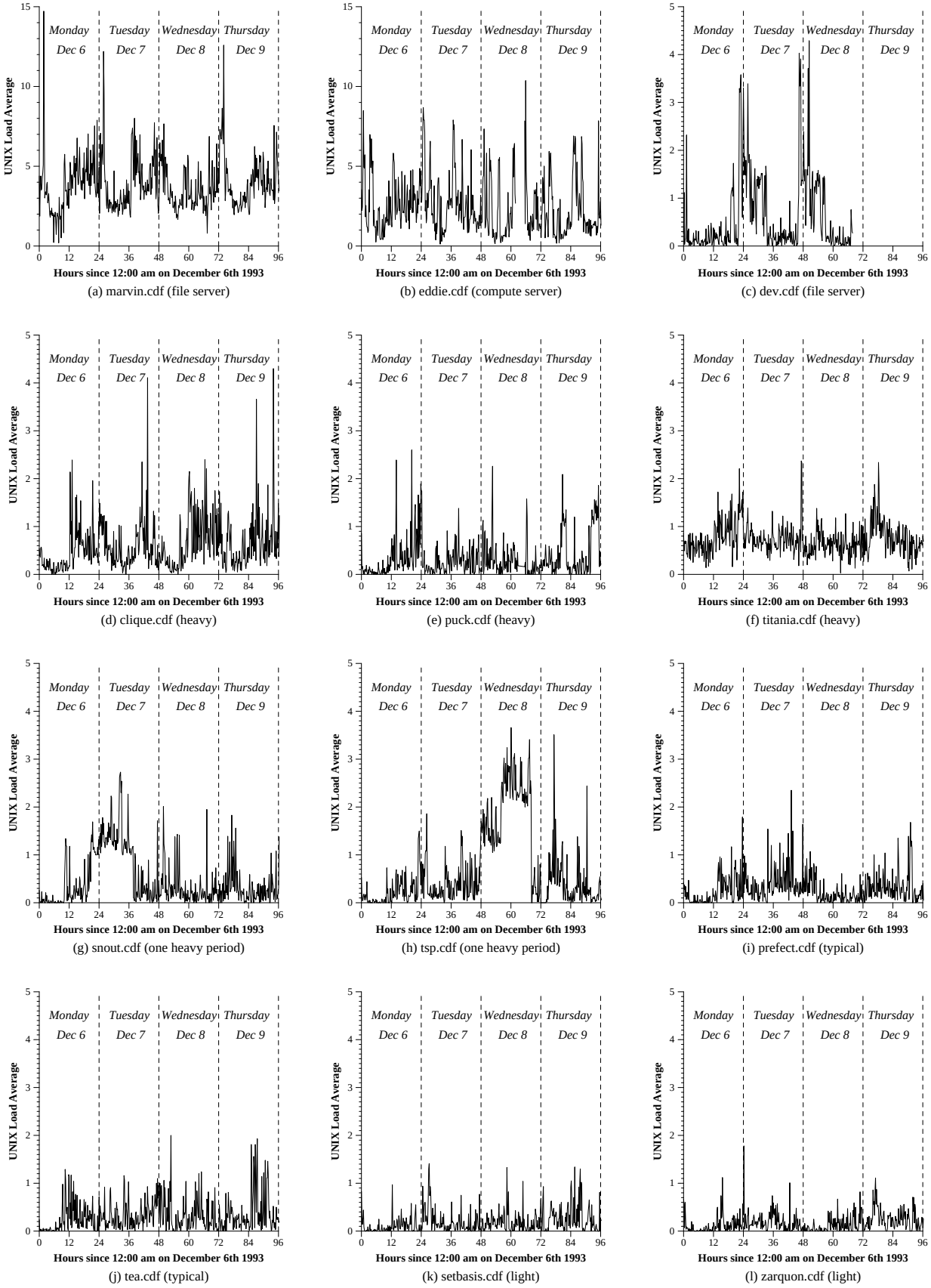
The daily peaks in eddie's load graph typically occur in the early afternoon. This is the period when eddie had the greatest number of users (approximately 30) logged in each day. As eddie's function in the CDF system is to provide powerful computing services to the users, it is not surprising that the load on eddie was influenced by user activity. Increased user activity in the CDF system as a whole resulted in more requests to eddie's news daemon, thus contributing even more to an increaseed UNIX load average on eddie during periods of high user activity.

The graph shown for dev in Figure A4.1 (c) is missing data, as dev went down at 7:50 pm on Wednesday. The load average on dev was very high around midnight on Monday and Tuesday. These peaks correspond to the nightly level zero tape dumps that were done each weekday evening, and to the maintenance of the workstations' local disks that was done by the `earlymorn` script on marvin. There was less activity early Monday morning because the level zero tape dumps were only performed on weekday evenings.

The remaining graphs that are displayed in Figure A4.1 are intended to provide a representative sample of the workload. Graphs (d)-(f) are from hosts that had a heavy load. Graphs (g) and (h) have one extremely heavy period. Graphs (i) and (j) have a typical load, and graphs (k) and (l) have a light load. Comparing the heavy load graphs to those with lighter loads reveals a high degree of load variability in the CDF system.

User activity was responsible for the heavy load periods that are shown in graphs (g) and (h). There was a `turing` job started by user "a468chap" in the process accounting records for snout.cdf that started at 19:36:31 on Monday and finished at 13:39:07 on Tuesday. This corresponds to the period with a high UNIX load average in graph (g).

[1] There is a discontinuity in the graph for eddie on Wednesday afternoon, corresponding to the period when eddie was down. The `top` command was restarted manually, but unfortunately there is a four hour period for which no `top` data were collected on eddie.



(a) marvin.cdf (file server)
(b) eddie.cdf (compute server)
(c) dev.cdf (file server)
(d) clique.cdf (heavy)
(e) puck.cdf (heavy)
(f) titania.cdf (heavy)
(g) snout.cdf (one heavy period)
(h) tsp.cdf (one heavy period)
(i) prefect.cdf (typical)
(j) tea.cdf (typical)
(k) setbasis.cdf (light)
(l) zarquon.cdf (light)

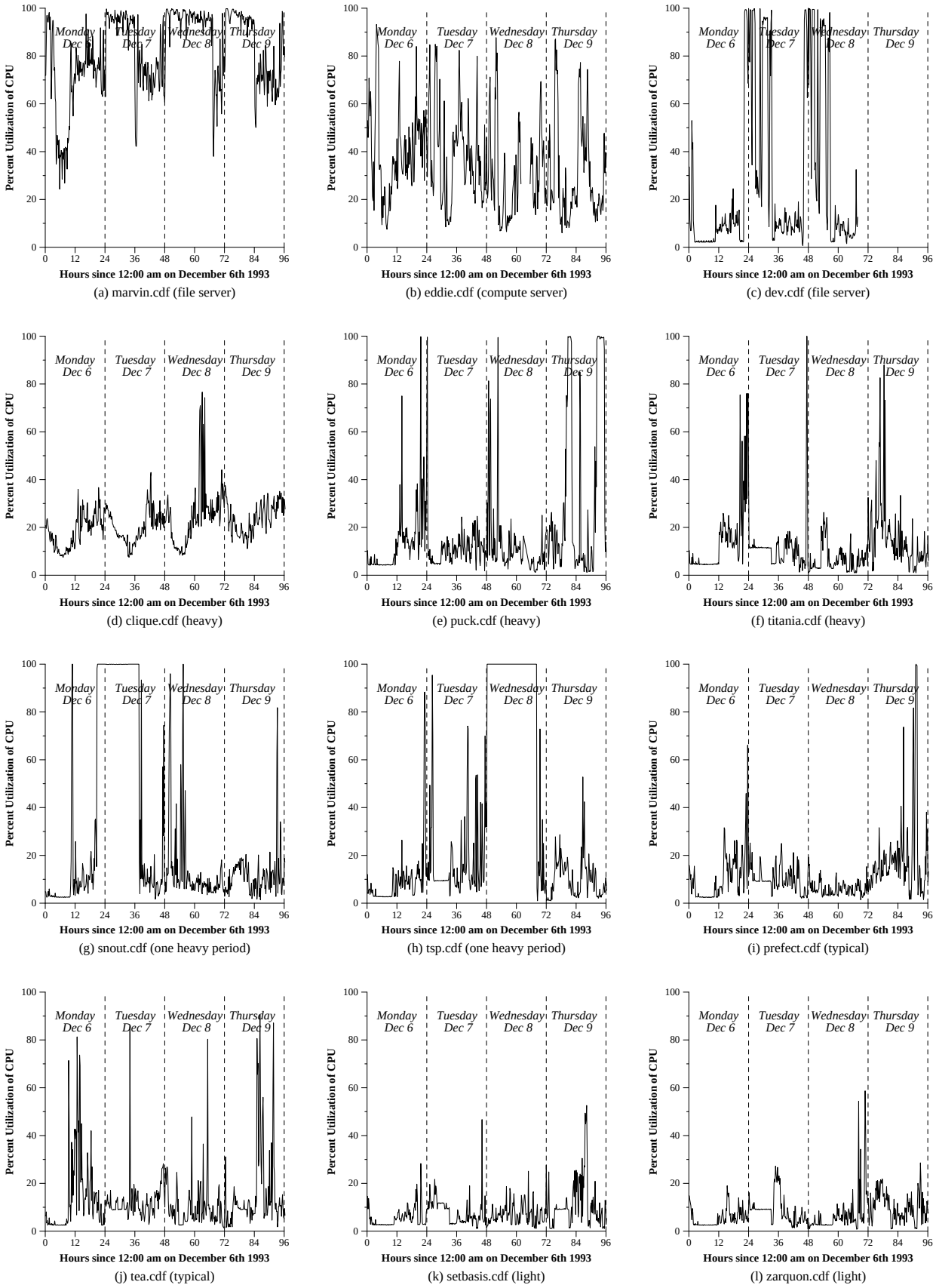
Figure A4.1: UNIX Load Average at 12 Minute Intervals on 12 CDF Hosts

Workstations zarquon (l) and setbasis (k) had particularly low loads over the entire four day collection period. It is difficult to determine why these hosts had such consistently low load averages. It may have been because these hosts usually had only one user logged in at a time. It may also have been because of the types of applications that users on these hosts were running. The consistently low loads on these hosts over the four day collection period is quite likely to have been somewhat coincidental.

A4.1.2 CPU Utilization

Figure A4.2 shows the percent utilization of the CPU for four days at 12 minute intervals on 12 different hosts. Subtracting the percent idle time that is provided by the `top` command, from 100% yields the percent utilization of the CPU.

In general, periods that had a high UNIX load average also had a high CPU utilization. In Figure A4.2, the "heavy" graphs (d)-(f) have a higher CPU utilization than the "light" graphs (k) and (l). The periods of heavy load in graphs A4.1 (g) and (h) have a 100% CPU utilization in graphs A4.2 (g) and (h).



(a) marvin.cdf (file server)
(b) eddie.cdf (compute server)
(c) dev.cdf (file server)
(d) clique.cdf (heavy)
(e) puck.cdf (heavy)
(f) titania.cdf (heavy)
(g) snout.cdf (one heavy period)
(h) tsp.cdf (one heavy period)
(i) prefect.cdf (typical)
(j) tea.cdf (typical)
(k) setbasis.cdf (light)
(l) zarquon.cdf (light)

Figure A4.2: CPU Percent Utilization at 12 Minute Intervals on 12 CDF Hosts

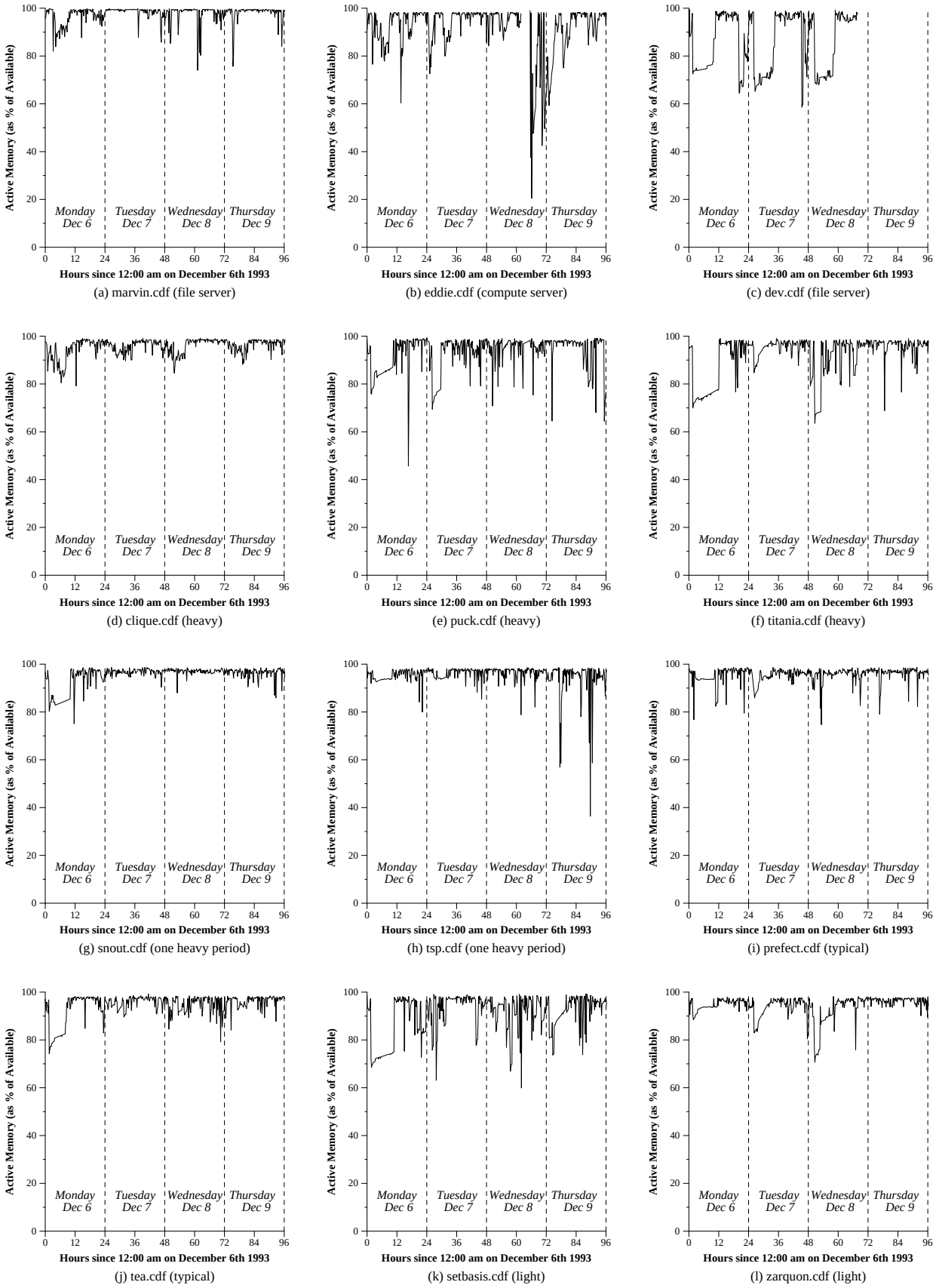
A4.1.3 Memory Usage

At 12 minute intervals, Figure A4.3 shows the memory that was active as a percentage of the available memory. Adding the free memory to the active memory gives the total amount of available memory on a host. The available memory plus the locked memory gives the total amount of memory on a host.

Graph (b) in Figure A4.3 shows an extreme drop in memory usage on eddie on Wednesday afternoon. This drop immediately follows the period when eddie was down. As the active memory was flushed when eddie was rebooted, more memory was free when the system restarted.

The host with the largest standard deviation for the percentage of active memory was dev, with a standard deviation of 12.15 (see Table 4.5 in Section 4.2.3). During the period from 2:00am until 11:00am each morning, the active memory usage on dev was particularly low. Usually periods that have low active memory have a correspondingly low load average and CPU utilization. During the 2:00am to 11:00am period on dev, however, the load average and CPU utilization were extremely high. This phenomenon is a consequence of CPU intensive `compress` command that was required to compress files before they were dumped to tape [DiM96]. Although this command consumes a lot of CPU, it has very low memory requirements.

In general, periods that had a high UNIX load average in Figure A4.1 and a high percent utilization of the CPU in Figure A4.2, had a large percentage of active memory in Figure A4.3.



(a) marvin.cdf (file server)
(b) eddie.cdf (compute server)
(c) dev.cdf (file server)
(d) clique.cdf (heavy)
(e) puck.cdf (heavy)
(f) titania.cdf (heavy)
(g) snout.cdf (one heavy period)
(h) tsp.cdf (one heavy period)
(i) prefect.cdf (typical)
(j) tea.cdf (typical)
(k) setbasis.cdf (light)
(l) zarquon.cdf (light)

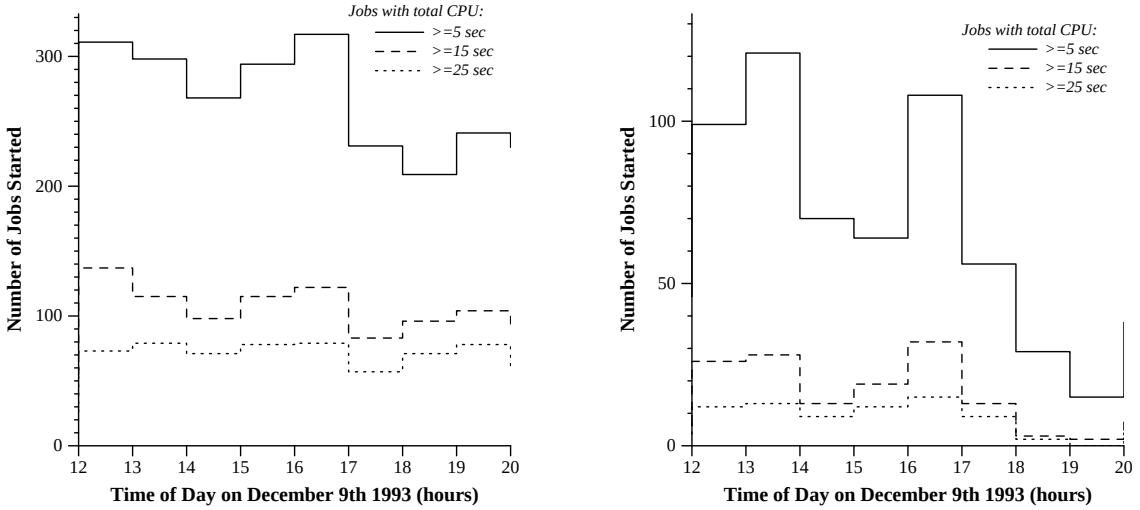
Figure A4.3: Active Memory Usage at 12 Minute Intervals on 12 CDF Hosts

A4.2 Selecting A Time Interval

In this section, we examine graphs for Thursday (from 12-8 pm) that were used to determine which time interval on Thursday was selected for the model. This discussion in this section follows from the discussion in Section 4.3.2 in the body of Chapter 4.

A4.2.1 Resource Intensive Jobs

The graphs in Figures A4.4 and A4.5 were generated from the process accounting records. Figure A4.4 uses a horizontal line to indicate the number of jobs given by only regular users [2] in a one hour interval requiring at least 5, 15, and 25 seconds of total CPU time. Figure A4.5 indicates the number of jobs in each one hour interval that read and/or wrote at least 25, 50, and 75 disk blocks. These jobs accounted for a significant portion of the resource usage in the system; consequently, their behaviour is of concern.



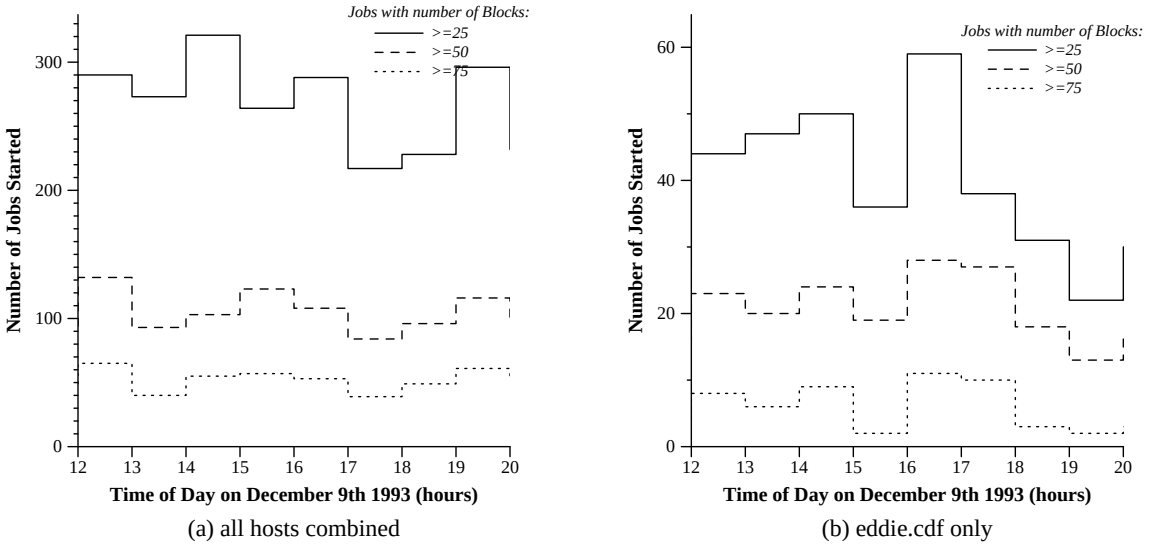
(a) all hosts combined: regular users

(a) eddie.cdf only: regular users

Figure A4.4: CPU-Intensive Job Creations by Regular Users

Figure A4.4 shows that the number of CPU-intensive jobs started on eddie and on all workstations is relatively constant from 12:00 pm until 5:00 pm, but it drops significantly after 5:00 pm. In Figure A4.5, a noticeable drop in the number of I/O-intensive jobs created per hour on eddie and on all workstations can be seen at 5:00 pm. On eddie there is also a pronounced drop in the graph at 3:00 pm.

[2] Regular users do not include system users, such as "root," "nobody," "sys," "news," and "daemon."

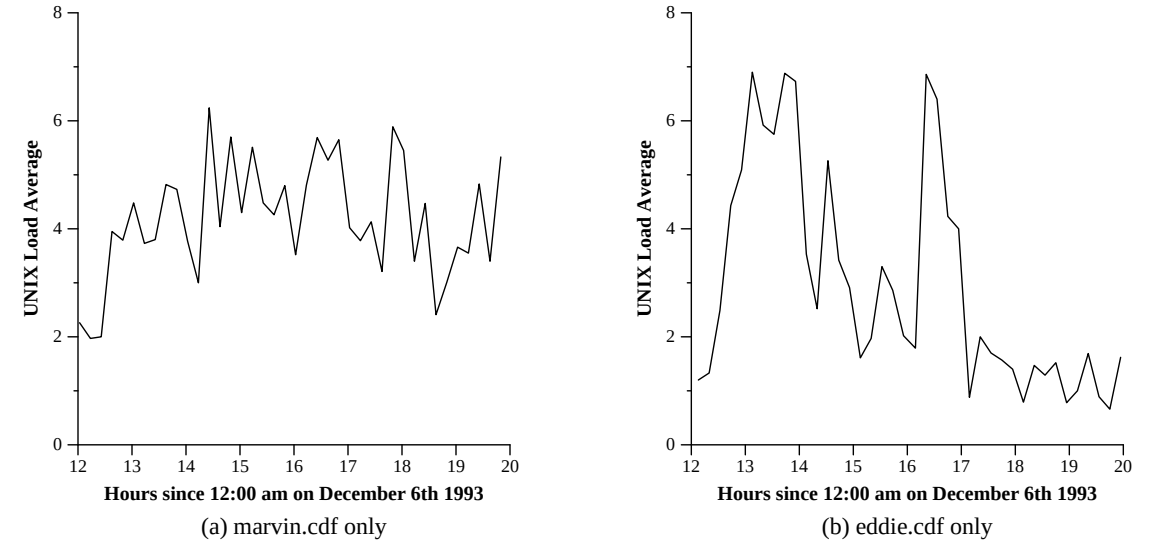


(a) all hosts combined (b) eddie.cdf only

Figure A4.5: I/O-Intensive Job Creations by Regular Users

A4.2.2 CPU Usage

The graphs in Figure A4.6 and A4.7 were generated from the `top` data. These graphs are the same as the graphs in Figure A4.1 and A4.2, except that only the interval from 12:00 pm until 8:00 pm on Thursday December 9^{th} is shown. Points are plotted at the midpoint of each 12 minute interval.

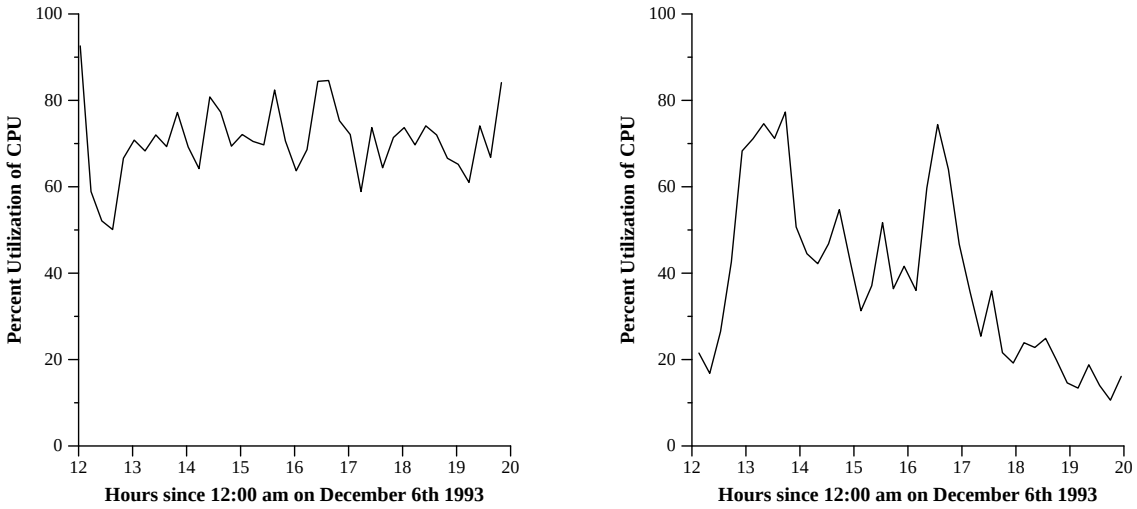


(a) marvin.cdf only (b) eddie.cdf only

Figure A4.6: UNIX Load Average at 12 Minute Intervals

The graphs in Figures A4.6 and A4.7 show the UNIX load average and percent utilization of the CPU at 12 minute intervals. Only the graphs for eddie and marvin are shown, as the load on these servers should be somewhat representative of the workstations that they

serve. The load and utilization on marvin are low between 12:00 pm and 1:00 pm, but from 1:00 pm until 8:00 pm, they remain relatively constant. There is much more variability in eddie's load and utilization. The load and utilization on eddie rise sharply between 12:00 pm and 1:00 pm, with noticeable peaks between 1:00 pm to 2:00 pm and between 4:00 pm to 5:00 pm.

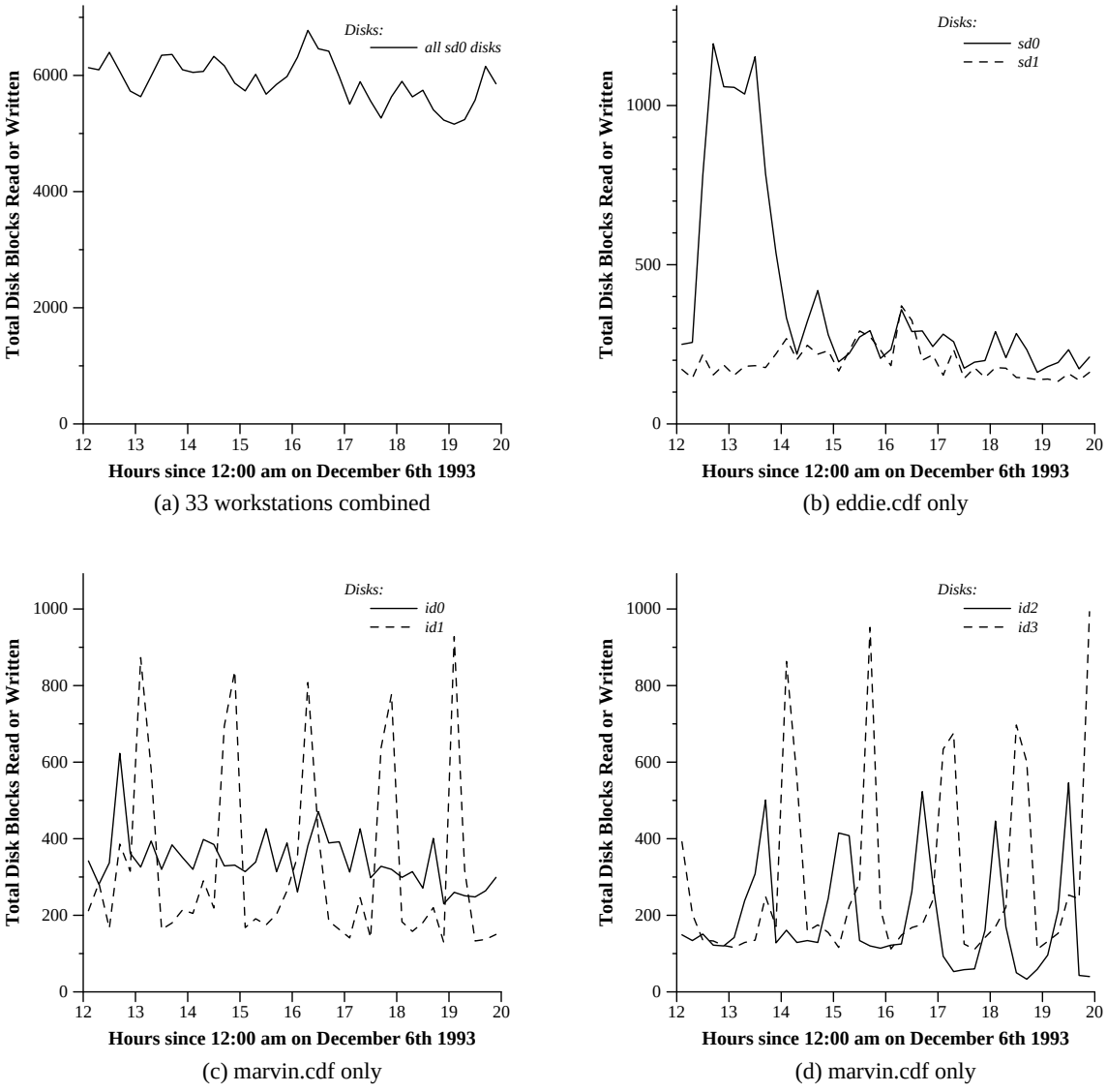


(a) marvin.cdf only (b) eddie.cdf only

Figure A4.7: CPU Utilization at 12 Minute Intervals

A4.2.3 Disk Usage

Figure A4.8 shows the total number of disk blocks that were read or written during each 12 minute interval on Thursday afternoon from 12:00 pm until 8:00 pm. These graphs were obtained by combining the 20 second interval `iostat` data over 12 minute intervals. Each point is plotted at the midpoint of the interval.



(a) 33 workstations combined

(b) eddie.cdf only

(c) marvin.cdf only

(d) marvin.cdf only

Figure A4.8: Disk Usage at 12 Minute Intervals

Graph (a) in Figure A4.8 shows the combined total number of disk blocks that were read or written on 33 client workstations. The line is relatively constant, but dips down to its lowest point at approximately 7:00 pm. Its highest point is at 4:30 pm.

Disk activity on eddie in Figure A4.8 (b), shows a high peak on the news `sd0` disk between 12:30 pm and 2:00 pm. This peak may have been because of an increased number of user requests during a period of heavy user activity.

Data for marvin's `id0` and `id1` disks are shown in Figure A4.8 (c); its `id2` and `id3` disks are shown in Figure A4.8 (d). The cyclic pattern that is evident at approximately 1.5 hour intervals is caused by the `backup` command that was run on marvin. This command cycled through five different partitions on marvin, backing up the data to two backup disks (`rf0` and `rf1`), which are not shown in these graphs. The disk activity on marvin's `id0` disk is not as severely influenced by the `backup` command because only the smaller mail partition was backed up on this disk. During the period from 12:00 pm until 8:00 pm, there was a noticeable incline on `id0` between 12:00 pm and 1:00 pm, followed by a fairly constant period, and then a gradual decline after approximately 5:00 pm.

A4.2.4 Number of Processes

The graph in Figure A4.9 shows the number of existing processes on eddie and marvin at 12 minute intervals. The number of existing processes on both of these hosts experienced a gradual decline between 1:00 pm and 8:00 pm. The number of existing processes on eddie reached a peak at approximately 4:45 pm and then dropped more sharply until 8:00 pm.

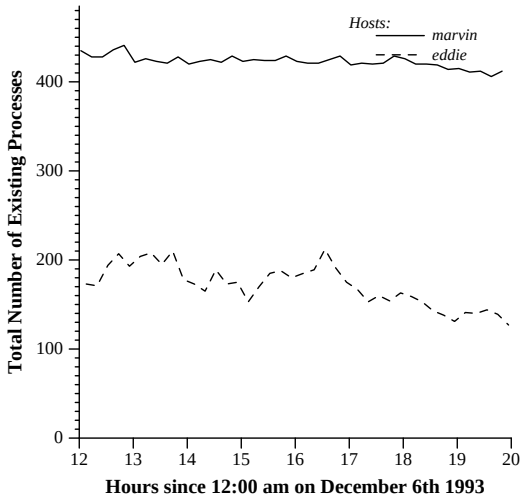


Figure A4.9: Number of Existing Processes at 12 Minute Intervals

A4.3 Analysis of Selected Interval

The tables in this section show resource usage statistics during the 1:00 pm to 5:00 pm interval on Thursday. In this section, we examine the performance of the disks, memory, CPUs, network, and NFS in the CDF environment. The discussion in this section provides performance insight that could be used by a system administrator to make decisions to improve the operation of the current system. This discussion is summarised in Section 4.4 of the body of Chapter 4.

Here is a brief description of each field that will be displayed in the tables for the `iostat` (Table A4.1), `vmstat` (Table A4.2), `top` (Table A4.3), and `netstat` (Table A4.5) commands.

iostat -D 20 command

disk	-	name of disk device
rps	-	number of disk blocks read per second
wps	-	number of disk blocks written per second
tps	-	number of disk blocks read/written per second
%read	-	percentage of disk reads: $rps \div tps * 100$
%util	-	percent utilization of disk

vmstat -S 20 command

r	-	# of processes in run queue (including process currently running)
b	-	# of processes blocked for resources (I/O, paging, etc.)
w	-	# of processes runnable but swapped out
si	-	Kbytes swapped in, per second
so	-	Kbytes swapped out, per second
fre	-	size of free list memory in Kbytes (pages of RAM ready to be used)
pi	-	Kbytes paged in, per second
po	-	Kbytes paged out, per second
sr	-	pages scanned by clock algorithm, per second (search for RAM to free)

top command

load	-	UNIX load average
%nice	-	nice CPU time percentage
%sys	-	system CPU time percentage
%user	-	user CPU time percentage
%util	-	percent utilization of the CPU

netstat -i 900 command

%error	-	percentage of packet errors
%colls	-	percentage of collisions on output packets
pack/30 sec	-	average number of packets per 30 second interval

A4.3.1 Disk

The tuning advice given in the Sun Performance Tuning Overview [Sun93] is that most systems usually have an overloaded disk. The system should be extensively examined using a command such as `iostat`, and even after tuning has been performed, it is advisable to recheck the system for an I/O bottleneck [3].

In this thesis, data from the `iostat -d 20` and `iostat -D 20` commands were collected. These data are summarised in Table A4.1. The most important field is **%util**. This field measures the amount of time that the disks take to respond to commands issued by the device driver, and how much time is idle between commands.

The tuning guide states that disks that are more than 30% busy should be carefully examined, as the most serious system bottleneck is usually an overloaded or slow disk. Table A4.1 shows that on average, the average utilization on 33 client workstation disks was 9.467%. With the exception of clique, whose local disk was 28.194% utilized on average, the client workstations do not have overloaded local disks. The I/O usage on clique is related to its virtual memory usage, which will be discussed further in the Memory subsection.

The two disks on dev were not highly utilized between 1:00 pm to 5:00 pm on Thursday, as these disks were primarily used during the early hours of the morning. The average utilization was only 0.067% on `sd0`, and 3.674% on `sd1`. Disk `sd0` stored the master copy of the client workstation disks; thus, no data were written to this disk, as is witnessed by the 100% read percentage in Table A4.1.

Eddie's `sd0` disk that stored the `/news` directory had an average utilization of 25.882%, while its `sd1` local disk had an average utilization of only 9.597%. The load on the news disk is quite high because of the frequent user requests during this period of high user activity. If the disk utilization on the news disk increased much more, it may be advisable to balance the load of this disk by adding another disk with an additional news partition.

Only data for the four IPI disks on marvin were collected by the `iostat` command. As outlined in Chapter 3, the `backup` command cycled through one directory on each of the `id0-id4` disks and the `sd0` disk on marvin. The `backup` command was extremely I/O intensive because it needed to check each of these directory for changes. The four `id` disks had average percent utilizations ranging from 13.420% to 23.843%. These averages may

[3] The bottleneck is defined as the resource that has the highest utilization in a system.

appear acceptable; however, the peak utilizations shown in the raw `iostat` data (which corresponded to the `backup` command usage) are a cause for concern. During 20 second intervals, the maximum **%util** shown for each of the four recorded disks on marvin were: 73.9% for `id0`; 98.8% for `id1`; 94.3% for `id2`; and 95.8% for `id3`. These maximum disk utilization percentages are considerably higher than the 30% utilization recommended by the tuning guide.

Perhaps the usefulness of the `backup` command should be evaluated to determine how necessary it really is. According to the `acctcom` records, the `backup` command was executed 12 times between 1:00 pm and 5:00 pm, reading or writing a total of 247452 disk blocks. This put a substantial load on the disk subsystem. Improved performance could be expected if the `backup` command was not executed. Perhaps as a compromise, it should be run once on each partition during the `earlymorn` backup session instead.

It should be noted that the actual number of read and write requests to the directories on marvin that had the Prestoserve accelerator installed is greater than the number of reads and writes that were propagated to the disks and were recorded by the `iostat` command. Had the 1 MB Prestoserve cache not been installed on marvin, the transfer rates and percent utilizations of marvin's disks are expected to have been even higher.

Since data were not collected on all eight of marvin's disks, it was impossible to evaluate the balance of the disk loads. Ideally files should be stored on the server disks so that incoming requests reference each disk at the same frequency. The most frequently requested files and file systems should be evenly distributed among the available disks.

The number of processes that are blocked while waiting for resources is another indicator of a disk bottleneck. This information is provided by the `vmstat` command and is shown in the **b** column in Table A4.2. When the number of processes that are blocked waiting for disk I/O is close to the number of processes in the run queue (the **r** column), a disk bottleneck is likely.

On average, 66 client workstations had an average of 0.083 processes blocked. Clique had the highest average number of blocked processes for the workstations (0.603). One `iostat` measurement showed that clique had 20 processes blocked waiting for I/O! The **b** average on eddie and marvin was quite high, but not compared to the **r** average on these hosts.

Host	disk	rps	wps	tps	%read	%util
clique	sd0	2.369	12.908	15.278	15.509	28.194
emerald	sd0	0.226	3.908	4.135	5.475	7.092
fjords	sd0	0.360	3.479	3.839	9.370	7.873
flollop	sd0	0.832	5.160	5.992	13.885	11.941
frogstar	sd0	1.318	3.862	5.181	25.442	11.505
frood	sd0	0.921	4.414	5.335	17.261	9.929
hoopy	sd0	1.160	5.536	6.696	17.320	12.118
indepset	sd0	0.243	3.433	3.676	6.611	6.358
janx	sd0	0.533	4.528	5.061	10.538	8.551
jasper	sd0	0.251	5.812	6.064	4.146	9.710
kclosure	sd0	0.385	3.601	3.986	9.652	7.091
knapsack	sd0	0.571	3.833	4.404	12.961	7.873
lysander	sd0	0.568	3.658	4.226	13.441	7.799
opal	sd0	0.549	8.319	8.868	6.186	18.521
prak	sd0	0.211	5.143	5.354	3.943	8.568
puck	sd0	0.189	3.871	4.060	4.653	6.956
quince	sd0	0.692	3.757	4.449	15.548	8.303
roosta	sd0	0.710	0.553	1.262	56.216	4.135
ruby	sd0	0.251	4.401	4.653	5.403	7.818
sapphire	sd0	0.040	3.697	3.737	1.078	5.882
sat	sd0	0.340	3.981	4.321	7.875	7.794
snout	sd0	0.500	3.446	3.946	12.672	7.015
snug	sd0	0.275	3.539	3.814	7.210	6.757
sofa	sd0	0.444	3.994	4.439	10.013	7.774
sundive	sd0	0.557	3.788	4.344	12.820	7.878
tea	sd0	2.449	5.342	7.790	31.432	15.600
tonnyx	sd0	0.424	4.910	5.333	7.943	9.527
topaz	sd0	0.037	3.503	3.540	1.059	5.587
towel	sd0	0.439	3.836	4.275	10.266	7.642
traal	sd0	0.419	4.508	4.928	8.512	8.593
trillian	sd0	0.590	4.156	4.746	12.438	8.644
tsp	sd0	2.074	5.290	7.364	28.159	16.182
zaphod	sd0	0.219	4.285	4.504	4.872	7.184
dev	sd0	0.011	0.000	0.011	100.000	0.067
	sd1	0.026	2.904	2.931	0.900	3.674
eddie	sd0	4.511	7.639	12.150	37.128	25.882
	sd1	1.240	5.106	6.346	19.545	9.597
marvin	id0	4.226	5.794	10.021	42.176	22.389
	id1	6.576	3.393	9.969	65.965	23.843
	id2	3.218	3.201	6.419	50.130	13.420
	id3	5.025	2.400	7.425	67.677	17.819
33 hosts	**33 disks**	**0.641**	**4.499**	**5.139**	**12.422**	**9.467**

Table A4.1: Averages of fields provided by the `iostat -D 20` command. These averages are for the 1-5 pm interval on Thursday. The final line shows the averages of the averages for the `sd0` disks on 33 client workstations.

A4.3.2 Memory

Each host in the CDF system had swap space allocated as a tmpfs partition on `/tmp` of the local disk [4]. This disk space served as virtual memory that allowed the hosts to run processes that were larger than their main memory (RAM) size. Each client workstation had 15 to 23 MB of swap space allocated on `/tmp`. The servers, which had larger memory sizes, also had larger swap spaces: 442 MB on eddie, 80 MB on marvin, and 38 MB on dev. On SunOS 4.X machines, the swap space is required to be at least as large as the amount of RAM.

A *paging* algorithm was used to manage and to allocate the physical memory pages that held the active parts of the virtual address space of processes running on a host. When processes are started or when a running process requires memory, pages of memory are allocated from the free page list. When the amount of free memory in the system becomes low, however, pages in memory that have not been used recently are added to the free list. A page daemon is woken up to scan the RAM to locate the page to be freed. If the page that is freed has been modified, it is paged out to disk.

Should the amount of available memory decrease even more, an algorithm is used to *swap* whole processes out to disk. This is referred to as a *hard swap*. A *soft swap* occurs when processes that have slept too long are automatically swapped out. Soft swaps are not included in `vmstat`'s **so** (swap out) statistic. Often only the private pages of a process are swapped out and put onto the free list. When a swap in (`vmstat`'s **si**) occurs, the pages may be reclaimed directly from the free list, without accessing the disk.

Although virtual memory makes it possible for systems to support multiple users and large programs, it is desirable to minimize its use. Accessing pages of virtual memory that are not currently paged in is much slower than accessing physical memory (RAM) because it generates a lot of disk activity. The memory that a process uses should be reduced, if possible, to fit into physical memory.

Averages for each `vmstat -S 20` field are shown in Table A4.2. If the **po** (page out) and **sr** (scan rate) columns from `vmstat` consistently show large values, this suggests that the host does not have enough main memory to support the applications that are currently running. Most client workstations had reasonably low page in and scan rate values. Clique,

[4] Marvin's swap space was distributed among 3 disks (`id1`, `id2`, `id3`).

however, had exceptionally high values for both of these statistics. The high average page out value of 1.950 KB/sec on clique indicates that pages belonging to other processes (or to the running process) frequently had to be written out to disk, to free up enough memory to run the current applications. The scanner was running continuously at a high rate (18.626 pages/second), as it searched for RAM pages to free.

The swap out field indicates when entire processes needed to be swapped out to disk. Again, most client workstations had very low **so** values. This is likely because most processes had short run times and low resource usage. The **so** values may have also been low because soft swaps were not recorded; consequently, processes that were swapped out because they slept for a long period of time were not recorded. Clique had the highest swap out rate of .0083 KB/second. This suggests that occasionally other processes were forced out of main memory, to accommodate the demands of the memory-intensive process that was running on clique. The process that caused the paging activity on clique could not be determined, as clique's process accounting records terminated at 6:00 am on Thursday morning, possibly due to a memory shortage.

Swap in and page in values are not indicative of the state of virtual memory in a system. The normal behaviour of initially loading a process into memory causes these fields to be incremented. Excessive values for these fields, supplemented with high page out and scan rate values could forecast a problem.

To determine the rate of paging on marvin and eddie, Adrian Cockcroft's *RAM Demand Rule* was used [Coc95]. This rule uses an approximation to idle page residence time (iprs), and is calculated as follows:

$$iprs = \frac{N_p \div 4}{sr}, \tag{A4.1}$$

where the number of pages in RAM, N_p, is determined by dividing the memory size by the page size (4 KB, except 8 KB on marvin), and **sr** is the `vmstat` scan rate. If the **iprs** is lower than 40 seconds, it indicates that RAM is low.

As expected, the lowest **iprs** value of 41.057 was found on clique. Both marvin and eddie had very high **iprs** values, indicating that there was not a RAM shortage on these hosts. The calculated **iprs** values are shown in the last column in Table A4.2.

	procs			swap		page				
Host	r	b	w	si	so	fre	pi	po	sr	iprs
agrajag	0.176	0.106	0.000	14.299	0.0000	314.078	4.406	0.144	2.858	267.551
amethyst	0.244	0.035	0.000	13.778	0.0000	438.578	0.872	0.000	0.408	2499.796
anjie	0.101	0.071	0.000	12.947	0.0000	376.328	2.022	0.000	1.387	551.171
babel	0.382	0.037	0.003	12.681	0.0000	378.806	1.756	0.000	1.175	650.851
benjy	0.158	0.050	0.001	10.743	0.0028	318.028	2.811	0.011	1.949	392.459
clique	0.421	0.603	0.010	17.771	0.0083	289.333	15.411	1.950	18.626	41.057
dent	0.126	0.071	0.006	12.403	0.0000	306.017	2.217	0.000	1.944	393.300
dev	0.211	0.014	0.001	17.465	0.0000	846.222	0.161	0.133	0.143	10714.369
diamond	1.203	0.046	0.000	15.497	0.0000	823.500	0.911	0.000	0.400	2551.875
eddie	**3.638**	**0.433**	**0.010**	**60.781**	**0.0000**	**4075.756**	**18.667**	**2.906**	**4.257**	**1918.219**
emerald	0.317	0.068	0.000	13.125	0.0000	501.589	1.378	0.000	0.735	1389.301
fjords	0.175	0.056	0.000	9.504	0.0014	343.439	2.400	0.000	1.887	405.166
flollop	0.340	0.110	0.003	16.249	0.0000	436.961	3.950	0.000	3.211	238.157
flute	0.210	0.058	0.003	14.471	0.0028	385.544	3.533	0.083	2.532	302.041
frogstar	0.200	0.115	0.003	13.169	0.0028	473.611	9.389	0.194	6.133	124.687
frood	0.142	0.115	0.000	12.938	0.0014	344.272	9.033	0.261	5.028	152.105
garnet	0.469	0.114	0.003	17.014	0.0014	617.944	9.378	0.261	5.525	184.751
hactar	0.121	0.032	0.000	11.192	0.0014	372.700	1.633	0.000	0.969	788.854
hcircuit	0.258	0.065	0.001	17.339	0.0014	296.922	2.667	0.011	2.442	313.208
hermia	0.147	0.067	0.000	11.963	0.0014	418.478	3.361	0.006	1.933	395.560
hoopy	0.222	0.142	0.006	19.919	0.0000	372.950	6.978	0.050	5.965	128.200
hrung	0.089	0.072	0.000	10.631	0.0014	314.756	5.561	0.150	3.797	201.397
indepset	0.174	0.031	0.000	12.282	0.0000	362.361	1.322	0.000	1.007	759.476
jade	0.440	0.074	0.003	14.729	0.0014	398.972	2.367	0.172	1.613	633.023
janx	0.165	0.054	0.000	14.629	0.0000	369.061	3.267	0.006	2.596	294.607
jasper	0.310	0.051	0.001	14.382	0.0000	2129.228	2.428	0.378	1.606	635.761
jeltz	0.163	0.090	0.001	19.354	0.0000	404.594	2.972	0.017	2.856	267.811
jynnan	0.078	0.026	0.000	10.469	0.0000	522.833	0.883	0.000	0.251	4060.442
kclosure	0.107	0.054	0.000	12.329	0.0000	358.044	1.717	0.011	1.386	551.723
kcolour	0.199	0.067	0.000	16.962	0.0000	338.994	2.083	0.000	2.293	333.507
killozap	0.239	0.099	0.001	13.781	0.0000	455.961	3.239	0.144	2.276	335.949
knapsack	0.151	0.068	0.001	11.510	0.0000	335.839	3.172	0.033	2.243	340.941
krikkit	0.188	0.060	0.000	13.363	0.0000	318.100	2.839	0.067	1.901	402.206
lysander	0.103	0.065	0.001	12.161	0.0000	369.422	2.839	0.044	2.276	335.949
marvin	**3.200**	**1.126**	**0.000**	**0.000**	**0.0000**	**745.111**	**71.811**	**2.378**	**0.706**	**4351.181**
matching	0.264	0.104	0.001	18.029	0.0000	364.017	4.044	0.017	3.035	252.000
mstlim	0.285	0.093	0.001	16.439	0.0000	423.200	3.117	0.011	2.260	338.427
oberon	0.293	0.068	0.003	11.596	0.0014	1153.750	3.878	0.122	2.917	349.971
opal	0.342	0.181	0.000	13.944	0.0028	569.478	4.417	0.250	2.422	421.411
planar	0.111	0.044	0.000	14.414	0.0000	376.589	2.433	0.000	1.285	595.265
prak	0.121	0.053	0.000	16.536	0.0000	311.356	1.417	0.000	1.199	638.030
prefect	0.292	0.126	0.001	12.715	0.0000	375.067	9.089	0.283	5.961	128.290
prosser	0.235	0.085	0.001	13.406	0.0028	349.767	6.894	0.028	5.478	139.610
puck	0.219	0.043	0.001	10.908	0.0014	889.261	1.489	0.078	0.644	1583.922
quince	0.118	0.075	0.000	13.625	0.0014	357.300	2.839	0.028	2.658	287.680
roosta	0.290	0.046	0.000	15.393	0.0000	413.767	4.906	1.339	4.189	182.566
ruby	0.218	0.051	0.003	13.667	0.0000	447.617	1.706	0.050	0.782	1305.400
sapphire	0.058	0.015	0.000	9.931	0.0000	415.556	0.744	0.000	0.285	3585.073
sass	0.201	0.113	0.001	12.882	0.0000	417.406	5.183	0.039	3.211	238.157
sat	0.212	0.049	0.001	13.369	0.0000	494.250	2.817	0.150	1.754	581.900
setbasis	0.353	0.160	0.003	16.126	0.0056	1022.428	6.000	0.522	4.881	209.146
snout	0.160	0.056	0.000	10.129	0.0014	365.900	3.289	0.033	2.233	342.425
snug	0.126	0.044	0.000	12.911	0.0000	355.267	1.944	0.128	1.394	548.426
sofa	0.117	0.060	0.000	12.296	0.0000	319.494	2.594	0.000	1.767	432.877
sundive	0.125	0.065	0.000	11.325	0.0000	440.367	2.894	0.061	1.972	387.761
tea	0.483	0.167	0.004	17.364	0.0028	384.911	13.783	0.128	8.797	86.931
titania	0.624	0.065	0.001	14.269	0.0000	614.617	2.544	0.011	1.636	623.888
tonnyx	0.214	0.064	0.000	16.911	0.0014	659.017	3.122	0.039	1.672	610.415
topaz	0.083	0.019	0.000	11.271	0.0000	1226.889	0.761	0.000	0.178	5741.719
towel	0.125	0.062	0.000	9.865	0.0069	647.989	5.628	0.806	2.029	503.039
traal	0.169	0.094	0.001	14.743	0.0000	533.172	1.972	0.039	1.082	943.440
trillian	0.207	0.057	0.000	12.719	0.0028	400.950	3.183	0.033	2.753	277.810
tsp	0.229	0.283	0.001	19.868	0.0014	550.972	14.006	1.161	10.036	76.200
vcover	0.419	0.090	0.001	16.478	0.0014	534.472	4.017	0.411	2.354	433.593
ws111	0.068	0.033	0.001	17.528	0.0000	2898.283	1.667	0.000	0.104	14714.400
x3cover	0.244	0.057	0.000	12.037	0.0000	1017.744	1.783	0.083	1.006	1015.110
yooden	0.171	0.079	0.003	15.224	0.0028	376.033	5.006	0.078	3.503	218.327
zaphod	0.181	0.042	0.001	10.771	0.0028	301.583	1.417	0.000	1.549	493.830
zarquon	0.226	0.075	0.000	14.018	0.0000	300.100	2.039	0.006	1.881	406.662
66 hosts	**0.233**	**0.083**	**0.0012**	**13.883**	**0.0010**	**527.209**	**3.809**	**0.150**	**2.729**	**903.282**

Table A4.2: Averages of fields provided by the `vmstat -S 20` command. These averages are for the 1-5 pm interval on Thursday. The final line shows the averages of the averages for 66 client workstations.

A4.3.3 CPU

If the CPU utilization is always greater than approximately 80%, the host may have an overloaded CPU. If the system is constrained by the CPU, possible solutions would be to get a faster CPU, to spread the workload across multiple CPUs (if available), or to somehow reduce the CPU requirement (load sharing, reschedule load to non-peak hours, improve efficiency of applications).

The graphs in Figure A4.2 showed that the utilization on most hosts was usually lower than 80% over the full data collection period. These graphs show that marvin and dev had periods of high CPU utilization during the early hours of the morning. Other hosts had bursts of heavy CPU utilization, primarily during the afternoon, but do not appear to be problematic.

The average CPU utilization percentage for each host during the 1:00 pm to 5:00 pm interval on Thursday is shown by the **%util** column in Table A4.3. The average of the average CPU utilization percentage for 59 client workstations was quite low at only 12.33%. The average for this period was 52.74% on eddie and 73.02% on marvin. This consistently high CPU utilization on marvin suggests that marvin had an overworked CPU.

Client workstation, diamond, had a 100% CPU utilization throughout the 1:00 pm to 5:00 pm interval. The percentage of user CPU time accounted for 95.28% of this total. Diamond was the only client workstation that had an average **r** (running processes) value that was greater than one. Examination of its process accounting records revealed a process called `auto` that was started by user g1howard at 21:53:46 on Monday night and did not complete until 18:17:44 on Thursday night. This process consumed a total of 30076 seconds of CPU time, 97% of which was user CPU time. Since `auto` is not the name of a standard command, it is difficult to determine the nature of this command. The command was not very I/O intensive, as it read or wrote only a total of 1141 disk blocks during its 68 hour duration; it was not memory intensive as low page out and scan rates were recorded for this host by `vmstat`.

High percentages of system CPU time indicate that CPU cycles are being expended in the kernel (OS code), whereas high percentages of user CPU time show that user processes are consuming CPU time. The Sun Performance Tuning Overview suggests that there may be a problem if there is more system CPU time than user CPU time and the machine is not an NFS server. In such cases, the source of system calls and interrupts should be determined

to pinpoint the problem.

In Table A4.3 both eddie and marvin had a larger system CPU time percentage than user CPU time percentage, as was expected. 21 client workstations, however, also had a larger system than user CPU time percentage. Of these hosts, only clique had more than a 2% difference. The high system CPU time percentage on clique was a result of the extra overhead required to perform excessive paging on this host.

Adrian Cockcroft's System Performance Monitoring column [Coc95] states that if the number of processes in the run queue determined by the `r` field of the `vmstat` command is lower than 3, there is no CPU problem. If the number is greater than 3, the time expended waiting in the run queue increases the interactive response time experienced by users. For hosts that have more than one processor, the number of running processes should be divided by the number of processors, based upon the approximation that each CPU takes one job off the run queue in each time slice. Thus, hosts that satisfy the following inequality do not have an overloaded CPU:

$$\frac{N_r}{N_c} < 3.0, \tag{A4.2}$$

where N_r is the number of processes in the run queue, and N_c is the number of CPUs.

Table A4.2 shows that on average, the client workstations had 0.233 processes in the run queue, which is significantly less than the acceptable 3 processes in the run queue. When the 4 processors on eddie are taken into consideration, eddie had an average of 0.91 processes in the run queue. Marvin is the only host for which the average number of processes in the run queue does not satisfy the inequality in Equation A4.2. On average marvin had 3.20 jobs in the run queue, indicating a potential CPU bottleneck. More CPU power should be added to this machine.

		CPU			
Host	load	%nice	%sys	%user	%util
agrajag	0.18	0.70	2.94	2.11	5.76
amethyst	0.27	0.26	3.60	5.21	9.07
anjie	0.12	0.02	2.78	2.56	5.40
benjy	0.17	0.24	2.71	2.81	5.76
clique	0.98	7.91	14.62	2.46	25.00
dent	0.24	1.03	3.77	5.89	10.73
diamond	1.19	0.01	4.67	95.28	100.00
eddie[5]	**4.24**	**0.15**	**16.48**	**10.64**	**52.74**
emerald	0.41	0.68	4.03	6.60	11.33
fjords	0.19	0.46	3.16	4.12	7.76
flollop	0.36	0.40	4.63	3.92	8.98
flute	0.27	1.22	4.15	4.67	10.06
frogstar	0.28	0.18	4.40	7.88	12.48
frood	0.19	0.03	3.54	3.71	7.28
garnet	0.53	2.36	6.36	7.68	16.38
hactar	0.14	0.01	2.94	3.75	6.74
hcircuit	0.31	0.27	4.29	6.60	11.17
hoopy	0.29	0.03	4.85	3.84	8.74
hrung	0.15	0.05	2.79	3.12	5.99
janx	0.26	0.69	4.07	5.99	10.79
jasper	0.42	0.97	4.35	5.12	10.44
jeltz	1.12	0.16	4.96	3.56	8.69
jynnan	0.11	0.01	3.00	2.60	5.66
kcolour	0.22	0.01	3.24	3.51	6.78
killozap	0.28	0.02	4.94	5.03	10.01
knapsack	0.16	0.09	3.60	4.32	8.05
krikkit	0.31	0.02	3.08	2.57	5.69
lysander	0.14	0.01	3.08	2.38	5.50
marvin	**4.63**	**0.50**	**65.28**	**7.22**	**73.02**
matching	0.40	8.96	4.64	5.74	19.37
mstlim	0.32	0.33	3.38	4.17	7.92
oberon	0.39	0.24	4.70	17.91	22.85
opal	0.49	1.86	8.44	9.14	19.46
planar	0.10	0.01	2.32	1.29	3.64
prak	0.16	0.38	3.59	3.90	7.88
prefect	0.36	1.49	6.75	11.87	20.12
prosser	0.39	0.03	5.07	5.98	11.09
puck	0.22	0.09	7.15	6.79	14.06
quince	0.14	0.22	3.93	3.43	7.62
roosta	0.35	0.04	4.08	5.24	9.37
ruby	0.25	0.03	2.72	1.76	4.53
sapphire	0.10	0.01	1.98	1.44	3.43
sass	0.31	3.15	4.60	3.21	10.99
sat	0.20	0.15	3.42	4.09	7.69
setbasis	0.49	0.03	5.46	18.69	24.21
snout	0.17	0.01	3.42	5.22	8.68
snug	0.12	0.28	2.47	1.60	4.37
sofa	0.12	0.02	3.64	5.12	8.79
sundive	0.21	0.02	3.60	4.01	7.64
tea	0.82	0.03	9.87	28.54	38.44
titania	0.73	0.11	4.62	6.16	10.91
tonnyx	0.30	0.09	5.32	4.88	10.31
topaz	0.09	0.01	2.15	3.31	5.52
towel	0.19	0.01	3.55	3.38	6.96
traal	0.24	0.35	3.77	4.62	8.75
trillian	0.31	0.01	5.53	3.46	9.03
tsp	0.40	0.24	6.75	8.93	15.94
vcover	0.54	5.08	6.33	14.57	26.00
yooden	0.21	0.03	4.36	6.69	11.10
zaphod	0.21	0.02	3.81	8.65	12.51
zarquon	0.27	0.32	3.57	4.27	8.19
59 hosts	**0.32**	**0.70**	**4.40**	**7.21**	**12.33**

Table A4.3: Averages of CPU fields provided by the `top` command.
These averages are for the 1-5 pm interval on Thursday. The final line shows the averages of the averages for 59 client workstations.

[5]The four CPUs on eddie also had an average of 25.48% for **%spin**.

A4.3.4 Network

Table A4.5 presents averages for `netstat -i` error percentages, collision percentages, and packet transfer rates for 43 hosts during the four hour interval. The last two fields in this table indicate the average number of input and output packets in 30 second intervals. The first three fields were calculated as follows:

$$\%input\ errors = input\ packet\ errors \div number\ of\ input\ packets \quad \text{(A4.3)}$$

$$\%output\ errors = output\ packet\ errors \div number\ of\ output\ packets \quad \text{(A4.4)}$$

$$\%output\ collisions = output\ collisions \div number\ of\ output\ packets \quad \text{(A4.5)}$$

The values for marvin were unreliable, as several negative values were recorded in the `netstat` output fields. In addition, the number of input packets on this host was always the same as the number of output packets. The **42 comb** line at the bottom of Table A4.5 gives the overall averages for the combined data of 42 hosts, excluding marvin.

Table A4.4 shows simple statistics for the `netstat` averages on the client workstations. The **40 hosts** line near the bottom of Table A4.5 matches the **mean** column in Table A4.4. The average percentages of input and output error averages on 40 client workstations were 0.00010% and 0.00031%, respectively. The highest input error percentage was 0.00170% on dev.cdf, and the highest output error percentage was 0.00916% on tea.cdf. According to [Ste90], the input error percentage should not be higher than 0.025%. A higher input error percentage may indicate that there are electrical problems with the transceiver or drop cable, or that packets are being dropped because of insufficient receive buffer space. The error percentage on CDF was significantly lower than 0.025%, indicating that no problems of this sort existed.

Measurement	min	max	mean	median	st. dev.
% input errors	0.00000	0.00166	0.00010	0.00000	0.00032
% output errors	0.00000	0.00916	0.00031	0.00000	0.00148
% output collisions	1.698	26.660	6.243	17.501	4.478
input packets/30 sec	58.550	1185.727	217.792	141.213	176.777
output packets/30 sec	2.475	1154.375	54.721	11.688	180.208

Table A4.4: Statistics of `netstat` averages for 40 client workstations

A network collision occurs when two hosts try to transmit on the Ethernet at the

same time. The combined average percentage of collisions for 42 hosts in the CDF system was 2.407%. An Ethernet that has less than 5% of its packets that result in collisions is considered lightly loaded; correspondingly, the network does not appear to be the bottleneck of the CDF system.

It is nonetheless interesting that a few hosts had extremely high collision percentages. The average collision percentage was 26.66% (530 collisions in 1988 packets) on sapphire, 17.50% (1196 collisions in 6834 packets) on jasper, and 12.19% on amethyst (403 collisions in 3307 packets). [Sun90] suggests that client workstations that have high collision rates should be examined for problems such as a bad transceiver, transceiver cable, or connection. If several clients in one area have abnormally high collision rates, [Sun90] states that the network topology and nearby cabling hardware should be examined. It seems more than coincidental that these three workstations with the highest collision rates were all on the same section of the Ethernet cable. All three hosts were in room 203, and attached to the `cdf-ether` Ethernet.

To determine if there were ongoing problems on this cable in room 203, the `netstat -i` records [6] that were collected by the static one time collection script were examined. Instead of finding a problem with the workstations in room 203, a more pronounced problem was indicated by the higher collision rates for the client workstations in room 201 north. The 6 workstations with the highest collision rates were all in room 201 north. The remaining 2 workstations in room 201 north had the 10^{th} and 12^{th} highest collision rates. The cause of these high collision rates in room 201 north is not known, and is beyond the scope of this study.

The number of input and the number of output packets per 30 second interval in Table A4.5 indicate which hosts were responsible for the traffic on the network. All hosts received more packets from the Ethernet than they put on the Ethernet. On average, the average input packet transfer rate for the 40 workstations was 217.792 packets per 30 seconds, and the average output transfer rate was 54.721 packets per 30 seconds. Eddie transferred and accepted packets at the fastest rate (1588.665 input packets/30 seconds, 1552.675 output packets/30 seconds) during the four hour interval. Of the client workstations, clique had the highest number of input (1185.727) and output (1154.375) packet

[6] When `netstat` is used without an interval parameter, statistics since the system was last booted are shown.

transmissions rates. This was likely a consequence of the thrashing that took place on clique.

There was no information collected about the time required by hardware and software to package and place the packets onto the Ethernet. Ideally, timing experiments should be performed to determine if this activity has an impact on the system. Such a study, nonetheless, was considered beyond the scope of this thesis, and was subsequently not carried out.

	%errors		%colls	packets/30 sec	
Host	**input**	**output**	**output**	**input**	**output**
agrajag	0.00000	0.00000	4.398	238.437	15.421
amethyst	0.00000	0.00000	12.186	58.615	3.521
benjy	0.00000	0.00000	7.960	237.306	7.648
clique	0.00035	0.00090	1.701	1185.727	1154.375
dev	0.00170	0.00000	1.926	122.742	78.656
diamond	0.00000	0.00000	7.849	64.706	10.073
eddie	0.00026	0.00107	1.838	1588.665	1552.675
emerald	0.00000	0.00000	3.116	174.575	188.448
flute	0.00000	0.00000	6.340	228.777	9.631
frogstar	0.00000	0.00000	2.938	260.794	34.150
hactar	0.00000	0.00000	7.140	227.363	7.004
hrung	0.00000	0.00000	6.317	227.615	6.669
indepset	0.00000	0.00000	6.949	58.550	10.577
janx	0.00090	0.00000	5.637	232.515	13.615
jasper	0.00000	0.00000	17.501	141.213	11.688
jeltz	0.00089	0.00000	2.812	233.246	3.158
kclosure	0.00000	0.00000	4.234	76.388	21.044
killozap	0.00000	0.00000	2.453	260.481	30.581
knapsack	0.00000	0.00000	4.376	69.590	16.488
krikkit	0.00000	0.00000	10.574	221.025	4.292
marvin	0.00000	0.00000	0.000	569.731	569.731
matching	0.00000	0.00000	8.450	84.408	13.021
mstlim	0.00000	0.00000	8.989	69.583	9.873
oberon	0.00000	0.00000	3.662	265.715	18.996
opal	0.00000	0.00000	3.975	147.731	84.950
planar	0.00000	0.00000	7.443	61.794	8.927
prak	0.00000	0.00000	9.868	250.940	7.294
prefect	0.00000	0.00000	3.931	259.812	22.837
puck	0.00000	0.00000	5.895	259.321	13.750
ruby	0.00000	0.00000	4.519	139.412	14.946
sapphire	0.00000	0.00000	26.660	70.783	2.475
sat	0.00000	0.00000	5.811	86.371	27.394
snout	0.00000	0.00000	4.572	226.183	6.533
snug	0.00000	0.00000	4.157	230.608	12.398
sofa	0.00000	0.00000	2.995	245.123	20.735
tea	0.00000	0.00916	1.698	439.115	159.123
titania	0.00000	0.00000	5.977	247.273	11.838
tonnyx	0.00000	0.00000	3.397	249.744	20.310
traal	0.00000	0.00000	5.126	246.898	16.277
vcover	0.00000	0.00000	5.549	160.848	27.998
ws111	0.00166	0.00253	4.180	251.329	82.471
yooden	0.00000	0.00000	1.899	295.073	52.535
zarquon	0.00000	0.00000	6.473	226.710	5.794
40 hosts	**0.00010**	**0.00031**	**6.243**	**217.792**	**54.721**
42 comb	**0.00018**	**0.00109**	**2.407**	**248.169**	**90.957**

Table A4.5: Averages of fields provided by the `netstat -i 900` command. These averages are for the 1-5 pm interval on Thursday. The **40 hosts** line shows averages of the averages for the 40 client workstations listed. The **42 comb** line shows averages calculated on the combined data for 42 hosts, not including marvin.

A4.3.5 NFS

NFS uses the Remote Procedure Call (RPC) facility to allow remote files to be accessed in the same manner as local files. The RPC facility translates local calls into requests for the file system on the remote host. When the client makes a request to the server and the server does not respond, the request times out on the client, and a retransmission must be performed. Excessive retransmissions cause network performance degradation and may indicate overloaded servers or unreliable components of the Ethernet. The `nfsstat` command can be used to differentiate between these two problems.

The first problem, an overloaded NFS server, is indicated by a *retransmission rate* that is greater than 5%. The busier the server, the more likely that the timeout interval will expire before the server gets a chance to handle the request.[7] The retransmission rate R can be calculated from the `nfsstat` output as follows:

$$R = \frac{timeout}{calls} \times 100. \tag{A4.6}$$

Average retransmission rates for 45 client workstations in the CDF system ranged from 0.0000% to 0.0396% between 1:00 pm to 5:00 pm on Thursday. The average retransmission rate average was 0.0061% for these 45 client workstations. There were few timeouts, indicating that the server was able to keep up with the requests generated by the client workstations, without any difficulty.

The second problem, a busy network, can be investigated by examining the **badxid** and **timeout** fields provided by the `nfsstat` command. The **badxid** counter indicates that a duplicate acknowledgement was received for a single request. If the network is poor, requests that time out do not get through at all (and thus are never acknowledged); consequently, the **badxid** value will be close to zero. In this case, the difference between the number of timeouts and the number of duplicate acknowledgements (**badxid**) will be large. When the **badxid** value is close to zero, the server does not see the original request transmission(s) because packets are lost in transit. The *Unacknowledged Request Rate U* is calculated as follows:

$$U = \frac{timeout - badxid}{calls} \times 100. \tag{A4.7}$$

[7] Busy NFS file servers can be caused by many factors, including overloaded Ethernet drivers, high CPU utilization, slow disks, an inadequate number of disk controllers, unbalanced disk loads, a small inode cache, and fragmented file systems.

When the unacknowledged request rate is greater than 5%, a network problem is likely. The unacknowledged request rate was very low in the CDF system (0.0058%), indicating that requests were rarely dropped because of network problems.

The `nfsstat -c` command also provides the NFS operation type of each call. The operation mix will vary significantly depending on the type of workload. The operation mix should be examined, as it can provide insight into the NFS performance. Table A4.6 provides average percentages of each NFS operation during the four hour period; the combined average for 45 client workstations, as well as the averages for dev and eddie are shown.[8]

The percentage of operations that can be satisfied by the inode cache (**lookup**, **getattr**, **setattr**) was very high on all hosts. The combined average for 45 workstations was 65.792% of all calls. Clearly, this indicates the importance of ensuring that a sufficiently large inode cache is installed on the NFS server.

The usual bottleneck on NFS servers is the disk subsystem. The time required to process NFS requests (memory speed) is significantly less than the time required for disk operations; accordingly, the NFS performance is generally limited by disk requirements. Writes in SunOS 4.X are synchronous, meaning that the server will not send an acknowledgement until the write completes. A large *NFS write mix* (**write**, **create**, **remove**) can degrade performance. If the NFS write mix is greater than 5% (and there is a sufficient volume of NFS calls), the Sun Prestoserve is recommended to improve NFS performance [Sun90].

The average NFS write mix was 4.220% for 45 client workstations in the CDF system. The swap space on the local disks of the client workstations was instrumental in lowering the NFS write mix. Often requests were satisfied by the disk cache, without requiring an NFS call. The write mix was 8.239% on eddie and 10.491% on dev. While these mixes were higher than those of the client workstations, the Prestoserve on marvin should minimize the performance degradation that these high NFS write mixes could cause.

The percentage of NFS **read** operations was much lower on dev (1.474%) and eddie (5.006%) than on the 45 client workstations (17.680%). This difference is because eddie and dev had a larger subset of marvin's file systems installed on their local disks. The Prestoserve on marvin may contribute to increased throughput of read requests as well.

If the percentage of **readlink** operations is exceptionally high, it should be ensured that

[8]The NFS operation mix for the NFS server, marvin, is not shown because `nfsstat -s` should have been used instead of `nfsstat -c` to collect server-side NFS statistics for marvin.

the links are not to file systems mounted on other servers, as performance is greatly reduced if the request must be propagated to another server. The combined **readlink** percentage for 45 client workstations was 11.220%, but no problem is foreseen because the links were to file systems mounted on the source file server (marvin).

NFS Operation	**Percentage of NFS Calls**		
	45 hosts	**eddie**	**dev**
null	0.000	0.000	0.000
getattr	40.523	24.594	36.452
setattr	0.135	0.817	0.360
root	0.000	0.000	0.000
lookup	25.134	33.438	27.402
readlink	11.220	24.191	21.288
read	17.680	5.006	1.474
wrcache	0.000	0.000	0.000
write	3.461	4.998	8.286
create	0.537	2.325	1.703
remove	0.222	0.916	0.502
rename	0.071	0.438	0.153
link	0.041	0.206	0.175
symlink	0.000	0.003	0.000
mkdir	0.003	0.016	0.000
rmdir	0.001	0.002	0.000
readdir	0.971	3.049	2.205
fsstat	0.000	0.000	0.000

Table A4.6: Average NFS Operation Mix from `nfsstat -c` command.
These averages are for the 1-5 pm interval on Thursday. The **45 hosts** column shows combined averages for 45 client workstation hosts.

A4.4 Command Usage

In this section, we discuss the resource usage of commands that were started between 1:00 pm and 5:00 pm on Thursday December 9^{th}. This discussion follows from the discussion in Section 4.6.2 in the body of Chapter 4.

Table A4.7 shows the 30 most frequently used commands for system users, and Table A4.8 shows the 30 most frequently used commands for regular users. As expected, these commands with the largest number of occurrences did not place a heavy demand on the system's resources.

		Mean Resource Usage							
Command Name [9]	Num. Occ.	Real (sec)	CPU (sec)			Chars. Transfer	Disk R/W	Kcore (min)	Hog factor
			Sys	User	Total				
sh	10551	3.57	0.06	0.00	0.07	297.76	0.53	0.21	0.019
grep	4367	2.60	0.10	0.37	0.47	4196.72	0.13	2.35	0.180
wc	3221	3.24	0.06	0.02	0.08	469.24	0.11	0.23	0.025
ruptime	3120	3.12	0.51	0.09	0.60	14553.97	64.89	3.28	0.193
cut	3054	3.54	0.08	0.02	0.10	135.90	0.12	0.26	0.027
awk	1973	2.85	0.19	0.02	0.21	1595.86	0.16	0.96	0.074
expr	1518	0.14	0.10	0.01	0.11	102.54	0.17	0.27	0.813
#lpd	1130	12.93	0.32	0.02	0.34	23563.34	13.17	2.38	0.026
in.rlock	1036	10.66	0.08	0.02	0.10	839.04	1.13	0.29	0.009
hostname	923	0.09	0.06	0.00	0.06	10.30	0.06	0.07	0.634
ls	900	0.27	0.20	0.01	0.21	563.23	0.02	0.68	0.795
stty	898	0.23	0.19	0.01	0.20	254.50	0.00	0.50	0.869
egrep	871	0.34	0.17	0.01	0.18	1185.91	0.18	0.56	0.542
rsh	842	3.66	0.12	0.02	0.14	3916.45	0.07	0.67	0.038
cat	732	0.25	0.15	0.01	0.16	1212.54	0.22	0.34	0.630
date	718	0.26	0.07	0.02	0.09	860.54	0.76	0.25	0.337
#printquota	665	0.64	0.19	0.03	0.22	5919.19	1.28	1.58	0.344
rpc.ruse	654	0.48	0.18	0.09	0.27	10938.38	1.06	1.24	0.558
basename	649	0.20	0.09	0.01	0.10	101.10	0.24	0.23	0.520
utmpclean	629	0.30	0.06	0.02	0.08	1383.59	0.38	0.20	0.260
rm	628	0.22	0.11	0.01	0.12	96.00	3.61	0.24	0.549
in.identd	611	0.29	0.12	0.01	0.13	112847.49	0.01	0.89	0.460
sed	572	0.33	0.23	0.01	0.24	807.64	0.02	0.74	0.734
#nntpd	476	175.86	1.52	0.27	1.79	291081.04	20.55	16.85	0.010
in.rshd	451	22.90	0.29	0.06	0.34	15546.84	0.65	1.44	0.015
nslookup	438	0.41	0.28	0.01	0.29	375.00	0.00	1.47	0.714
#csh	429	6.33	0.93	0.24	1.17	793.72	2.32	6.71	0.184
#chown	375	1.26	0.13	0.02	0.15	810.84	36.06	0.71	0.117
tee	374	0.47	0.06	0.02	0.08	236.15	1.40	0.21	0.173
syslogpu	373	0.64	0.08	0.02	0.10	888.73	0.93	0.34	0.156

Table A4.7: Most Frequent Commands used by System Users

[9] A "#" was prepended to the command name if the command was executed with super-user privileges.

The most frequent system commands were typically UNIX file handling utilities. The top

Comm. Name	Num. Occ.	Mean Resource Usage							
		Real (sec)	CPU (sec)			Chars. Transfer	Disk R/W	Kcore (min)	Hog factor
			Sys	User	Total				
sh	6452	30.27	0.05	0.01	0.06	471.70	0.44	0.23	0.002
ls	2334	0.84	0.18	0.03	0.21	841.40	0.34	1.13	0.250
cpp	1165	0.60	0.14	0.08	0.23	37180.64	4.74	1.85	0.374
rm	1032	1.16	0.10	0.02	0.12	128.80	0.84	0.49	0.106
hostname	873	0.20	0.11	0.00	0.11	10.61	0.49	0.30	0.562
ld	760	4.46	0.54	0.40	0.93	483584.90	19.08	25.34	0.210
gcc	760	7.90	0.13	0.04	0.16	585.40	4.73	0.94	0.021
#tcsh	756	4126.14	3.14	0.57	3.70	29239.30	9.42	33.36	0.001
tcsh	721	64.81	0.14	0.02	0.16	882.04	0.37	1.67	0.003
vi	712	273.56	0.83	0.30	1.13	39992.25	16.14	12.24	0.004
as	681	1.33	0.24	0.63	0.87	56340.95	5.52	15.38	0.660
cc1	671	4.11	0.36	2.24	2.60	50801.29	5.10	87.27	0.632
clear	671	0.39	0.13	0.02	0.15	6533.92	0.49	0.67	0.395
more	640	37.07	0.25	0.05	0.30	16071.21	0.70	1.59	0.008
Mail	620	122.38	0.44	0.26	0.69	178148.33	12.19	6.19	0.006
stty	553	0.26	0.14	0.02	0.16	96.67	0.95	0.88	0.604
grep	530	2.14	0.10	0.05	0.15	14230.40	2.02	0.63	0.069
mesg	501	0.39	0.15	0.02	0.17	96.02	1.83	0.86	0.434
#lpq	481	1.04	0.19	0.09	0.28	13759.77	0.86	2.27	0.265
msgl	457	4.92	0.41	0.09	0.51	112494.95	7.35	3.56	0.103
cat	450	23.25	0.14	0.02	0.16	12816.71	1.02	0.49	0.007
whoami	420	0.54	0.26	0.02	0.29	709.10	0.44	0.97	0.533
#xterm	398	4618.17	3.23	1.52	4.75	361127.64	15.33	52.74	0.001
kill	385	0.31	0.11	0.01	0.13	96.00	0.94	0.57	0.415
xrdb	360	0.55	0.12	0.06	0.18	4576.69	0.02	1.13	0.323
quota	348	10.06	0.35	0.05	0.40	7917.80	7.19	2.96	0.040
make	343	15.51	0.30	0.11	0.41	8626.71	2.82	3.88	0.026
#lpr	340	3.12	0.45	0.30	0.75	90289.31	22.52	7.10	0.239
nm	335	0.27	0.07	0.07	0.15	47168.84	0.33	0.78	0.539
turing	335	452.91	2.09	1.54	3.63	71950.36	2.99	41.08	0.008

Table A4.8: Most Frequent Commands used by Regular Users

five system commands were part of the following six-command script that was executed by root on each workstation at 5 minute intervals: `ruptime, wc, cut, sh, grep, sh`. The other frequently used system commands were part of scripts or invocations of daemons.

The list of commands frequently given by regular users was significantly different than those commands given by system users. Commands for programming (`turing`), compiling (`gcc, cc1, ld, make, cpp`), assembling (`as, nm`), editing (`vi`), printer interface (`lpq, lpr`), and user communication (`Mail, msgl, mesg`) were most frequently used by regular users.

Table A4.9 shows the regular user commands that used the most total CPU time. For the majority of these commands, the high CPU usage is because of the long elapsed

times of the commands. The hog factor [10] was not particularly high for these commands, indicating that although they consumed a lot of CPU, it was distributed over an extended period of time. Processes like these are unlikely to merit remote execution in load sharing environments. Long running editors, rendering packages, programming environments, and window managers were among the commands that required a lot of processing power.

		Mean Resource Usage							
Comm. Name	Num. Occ.	Real (sec)	CPU (sec) Sys	User	Total	Characters Transfer	Disk R/W	Kcore (min)	Hog factor
saved_kc	11	2434.30	219.95	264.39	484.34	2076986.09	222.82	43014.06	0.199
top	21	1636.90	122.33	43.72	166.06	353787203.29	4.95	7711.35	0.101
term	3	5747.07	117.76	30.64	148.40	1755306.67	4.67	3541.03	0.026
oot	120	4328.15	29.72	45.30	75.02	5100477.99	44.66	4548.09	0.017
xv	33	1195.21	7.60	53.58	61.18	1938670.70	39.09	8287.43	0.051
rbp	1	6916.27	23.28	32.83	56.11	325440.00	47.00	2680.04	0.008
render	134	141.11	4.38	45.53	49.91	431708.87	4.69	2992.37	0.354
#matlab	6	1712.63	21.19	18.11	39.30	401600.00	59.50	3322.48	0.023
xlock	122	294.94	3.97	34.43	38.40	12084597.69	2.69	614.08	0.130
irc	4	2339.72	7.27	17.84	25.10	301778.00	12.25	384.32	0.011
emacs	57	1383.74	8.15	14.74	22.89	462819.79	34.72	805.60	0.017
txconn	4	3995.39	22.15	0.38	22.53	1624254.00	1.00	54.97	0.006
emacs-19	11	940.04	5.41	15.85	21.26	609605.82	70.27	836.01	0.023
xrwho	8	2836.68	17.30	3.18	20.48	819265.00	1047.12	223.37	0.007
textedit	5	2845.95	7.90	11.05	18.95	1411857.60	24.40	436.74	0.007
cmdtool	2	1748.26	6.30	12.29	18.59	656128.00	13.50	475.99	0.011
#screen	5	2485.45	16.62	1.12	17.74	413043.20	22.60	155.15	0.007
mwm	4	6114.82	12.53	4.38	16.91	1132584.00	35.75	389.20	0.003
renderde	25	505.00	1.33	15.52	16.85	307909.12	5.40	1316.58	0.033
xroach	6	1066.75	13.99	2.78	16.77	1123658.67	1.83	154.59	0.016
animator	4	68.03	7.35	8.93	16.28	2290622.00	5.00	669.85	0.239
.fvwmex	2	4610.66	10.42	4.08	14.50	1853440.00	11.50	133.56	0.003
kermit	12	256.22	13.45	0.67	14.13	102400.50	4.42	150.90	0.055
xrn	18	511.86	5.13	8.13	13.27	709678.67	22.00	1431.07	0.026

Table A4.9: High CPU Usage Commands used by Regular Users

The commands shown in Table A4.10 are the regular user commands that had the largest hog factor values. In general, commands with high hog factor values have the most potential for remote placement in load sharing policies because their processor requirements are high compared to their run time. Although the commands shown in Table A4.10 had the highest hog factor values, the average total CPU time required by these commands was quite low. This reaffirms our previous observation, in Section 4.6.1, that the workload in the CDF system is unsuitable for load sharing.

[10] The hog factor is the total CPU time divided by the elapsed time.

		Mean Resource Usage							
Comm. Name	Num. Occ.	Real (sec)	CPU (sec)			Chars. Transfer	Disk R/W	Kcore (min)	Hog factor
			Sys	User	Total				
sumt2	6	0.18	0.17	0.01	0.18	269.17	0.00	0.47	1.000
fpversion	4	8.33	0.14	7.58	7.72	236.25	2.25	44.87	0.927
canonhdr	2	0.17	0.15	0.01	0.16	476.00	0.50	0.43	0.914
getopt	5	0.11	0.09	0.01	0.10	126.60	0.20	0.26	0.895
consolet	330	0.08	0.05	0.02	0.07	100.00	0.01	0.17	0.892
t+toc.x	45	1.21	0.19	0.80	0.99	34065.60	2.31	20.60	0.820
ypwhich	4	0.24	0.18	0.01	0.20	107.00	1.50	0.60	0.804
tty	31	0.50	0.38	0.02	0.40	107.19	0.65	1.19	0.803
scanpars	65	1.26	0.12	0.83	0.95	28535.75	0.82	11.74	0.754
mknod	6	0.10	0.06	0.02	0.08	96.00	0.33	0.16	0.742
ccom	118	2.25	0.29	1.35	1.64	68521.73	3.48	24.62	0.730
write	4	0.26	0.16	0.02	0.18	2990.50	0.50	0.76	0.702
basename	66	0.13	0.07	0.02	0.09	102.33	0.24	0.21	0.701
uname	7	0.14	0.09	0.01	0.10	97.43	0.71	0.29	0.690
test	39	0.17	0.11	0.01	0.12	96.00	0.13	0.52	0.687
expr	287	0.16	0.09	0.02	0.11	97.88	0.06	0.30	0.686
users	4	0.37	0.24	0.01	0.25	3900.00	1.00	0.95	0.678
look	1	0.82	0.53	0.02	0.55	49272.00	14.00	1.06	0.671
ctime	1	0.18	0.12	0.00	0.12	176.00	1.00	0.36	0.667
calendar	1	0.12	0.08	0.00	0.08	985.00	1.00	0.24	0.667
as	681	1.33	0.24	0.63	0.87	56340.95	5.52	15.38	0.660
fsp_clie	1	0.20	0.13	0.00	0.13	141.00	1.00	0.63	0.650
cc1	671	4.11	0.36	2.24	2.60	50801.29	5.10	87.27	0.632
cg	3	1.53	0.23	0.72	0.96	210368.00	6.00	13.05	0.627

Table A4.10: High Hog Factor Commands used by Regular Users