above (the P*s reflect the proportional service requests that an IO device must sustain for optimal throughput as shown by Buzen [B7].)

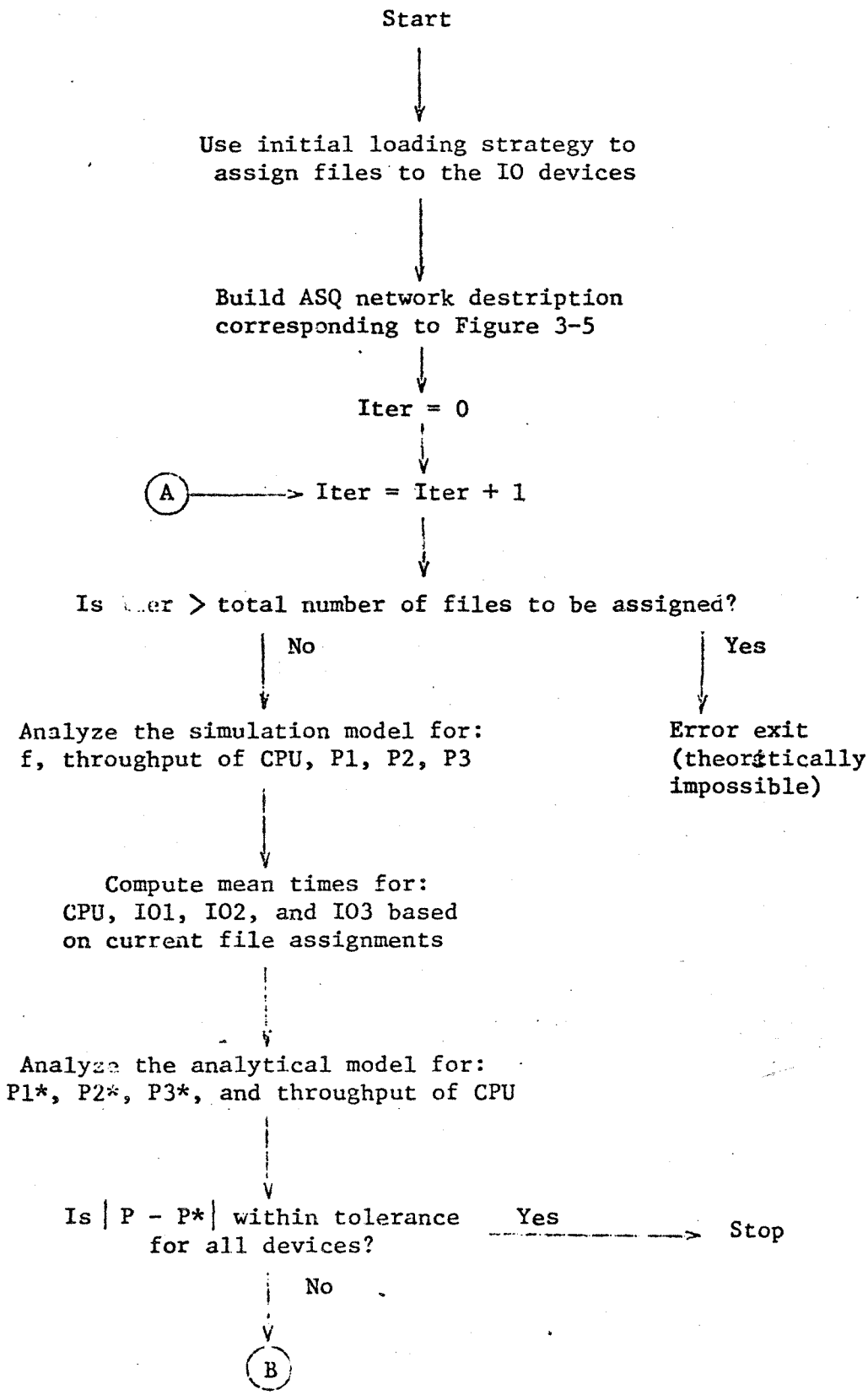
d. Terminating condition 1: Are the corresponding P and P* within tolerance? If yes, stop. Otherwise, continue.

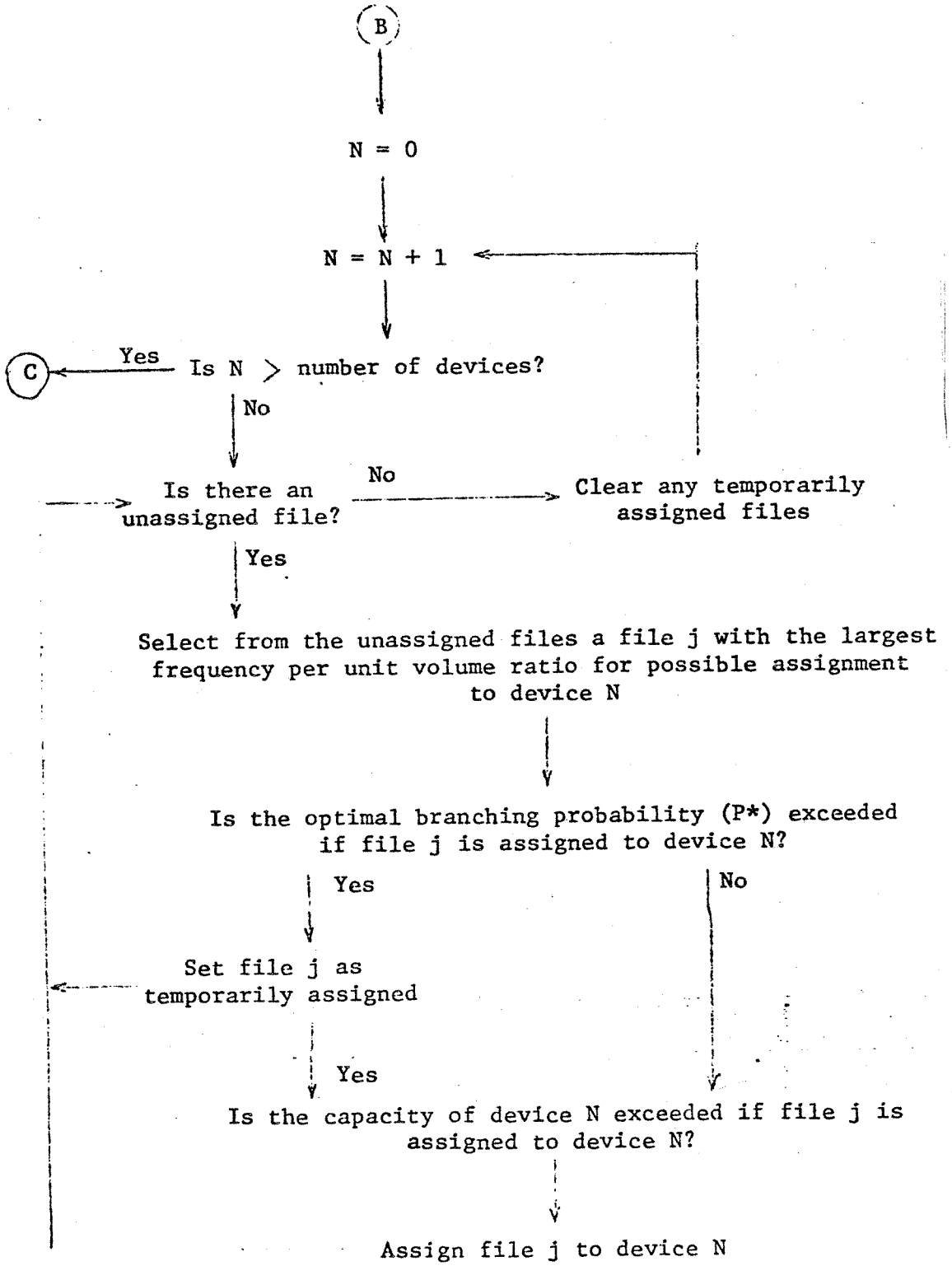
e. Reload the IO devices with the files whose frequency of loading sum to the corresponding P* computed via the branch and bound approximation described earlier, also recompute the mean services times.

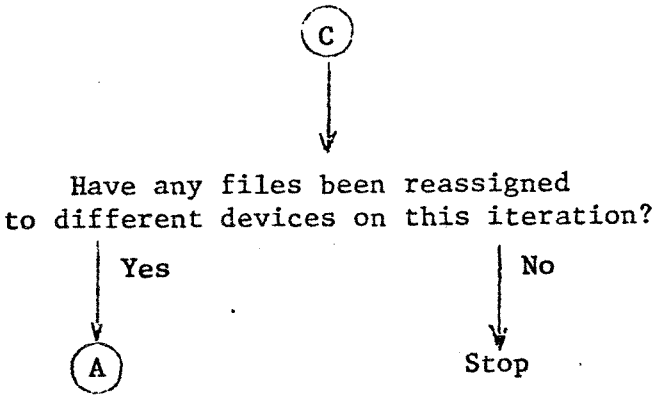
f. Terminating condition 2: Are any files reassigned to different devices? If no, stop. Otherwise go to b.

The simulation model refines the value of f and the system throughput while the analytical model refines the values of the Ps. Cooperation between the simulation and analytical models is useful in making the throughput analysis both feasible and tractible.

A more detailed description (than the above) of the iterative procedure is given in the following flowchart. The reader is cautioned not to lose his perspective.


Start
Use initial loading strategy to assign files to the IO devices
Build ASQ network destription corresponding to Figure 3-5
Iter = 0
A
Iter = Iter + 1
Is Iter > total number of files to be assigned?
No
Yes
Analyze the simulation model for: f, throughput of CPU, P1, P2, P3
Error exit (theoretically impossible)
Compute mean times for: CPU, IO1, IO2, and IO3 based on current file assignments
Analyze the analytical model for: P1*, P2*, P3*, and throughput of CPU
Is | P - P* | within tolerance for all devices?
Yes
Stop
No
B


B
N = 0
N = N + 1
C
Yes
Is N > number of devices?
No
Is there an unassigned file?
No
Clear any temporarily assigned files
Yes
Select from the unassigned files a file j with the largest frequency per unit volume ratio for possible assignment to device N
Is the optimal branching probability (P*) exceeded if file j is assigned to device N?
Yes
No
Set file j as temporarily assigned
Yes
Is the capacity of device N exceeded if file j is assigned to device N?
Assign file j to device N


C
Have any files been reassigned
to different devices on this iteration?
Yes
No
A
Stop

3.3.3 An Example

An example illustrating the above procedure is now given. It should be noted that the analysis technique considers the queuing of file requests and distributes them to the IO devices to reflect the proportion that each device must sustain for optimal throughput. The importance of this optimal distribution is emphasized by noting that the relative improvement in system throughput between the first and last iterations is 23%.

Some preliminary definitions are given before describing the example. System throughput is the throughput measured at the CPU since all requests flow through it. The activity profile is given in Figure 3-2. It consists of a) 15 program files, 13 of which are reusable (i.e., if a copy is already loaded when the file is requested, then that same copy is reused without having to reload a new copy), and b) 27 data files, 1 of which is reusable. When processing of a non-reusable file finishes, that file is deallocated from executable memory. When the processing of a reusable file finishes, that file remains in executable memory until it is replaced by another file which needs its memory. If a reusable or non-reusable file is a data file, its memory is deallocated only after it is rewritten to the appropriate device with a probability of 0.1. Note that the frequencies of requesting a file are unnormalized, and the volumes vary from 254 words to

5,630,400 words.

The degree of multiprogramming (i.e., the number of jobs located anywhere in the model) is five. The hardware characteristics of the system model topology shown in Figure 3-3 are the same as those described earlier except a) the CPU has a mean execution time/instruction of 1.0 microsecond, b) the capacity of the single executable memory is large enough to prevent memory queuing of the five largest records in the activity profile but small enough to prevent the entire activity profile from being permanently loaded into executable memory, c) the capacity of any of the three IO devices is large enough to load all files, and d) five channels are available for data transfer. These characteristics imply that no memory or channel queuing occurs in Figure 3-4. Also the system is load bound due to the speed of the CPU.

The initial strategy of loading the most frequently executed files (not the most frequently loaded) on the fastest IO device without overflowing its capacity loads all of the 42 files of the activity profile on IO1 as shown in Figure 3-6. The column headings of this figure are interpreted as follows: a) N is a sequential numbering of the files for convenience, b) J, ID, F, R, V are the parameters of the activity profile, c) ACTUAL REL FREQ is the normalized frequency of requesting a file, d) OBSERVED REL FREQ is the normalized frequency of observed file requests, e) OBSERVED OBJ REQ is the number of times each

III. OBJECT STATISTICS

A. LOGICAL MEMORY: (1: 1) MEMORY NAME: (FMEM ,IO1) MEMORY CAPACITY: (100000: 99000000)

N	J	ID	F	I	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ REQ	COMPLTED LOAD REQ	CURRENT LOCTN	CURRENT STATUS XEQ	LOD	LDG
1	41	D	5.0000	75	1056	3168	.2329677	.2286171	1144	1257	IO1	0	0	2
2	1	PR	3.1000	6000	2112	2112	.1444400	.1436851	719	1	FMEM	0	1	0
3	18	D	1.4200	304	304	516800	.0661628	.0611511	306	338	IO1	0	0	0
4	2	PR	1.2000	3000	6000	6000	.0559123	.0547562	274	1	FMEM	0	1	0
5	25	D	1.1300	40	46	5630400	.0526507	.0495604	248	263	IO1	0	0	1
6	29	D	.9120	30	33	600000	.0424933	.0437650	219	241	IO1	0	0	0
7	26	D	.8430	47	47	135360	.0392784	.0385691	193	217	IO1	0	0	0
8	17	D	.8200	1	2	264000	.0382067	.0351719	176	187	IO1	0	0	0
9	5	PR	.7600	2900	254	254	.0354111	.0355715	178	1	FMEM	0	1	0
10	37	D	.7500	7	7	54012	.0349452	.0357714	179	194	IO1	0	0	0
11	31	D	.5600	50	202	126000	.0260924	.0257794	129	146	IO1	0	0	0
12	19	D	.4920	3	6	90024	.0229240	.0207834	104	113	IO1	0	0	0
13	21	D	.4920	10	2000	113154	.0229240	.0263789	132	151	IO1	0	0	0
14	7	PR	.3200	2400	2752	2752	.0149099	.0169864	85	1	FMEM	0	1	0
15	6	PR	.3200	2900	3200	3200	.0149099	.0157874	79	1	FMEM	0	1	0
16	23	D	.2960	27	27	216000	.0137917	.0139888	70	76	IO1	0	0	0
17	14	P	.2960	10	64	21504	.0137917	.0107914	54	54	IO1	0	0	0
18	16	D	.2680	10	150	24000	.0124871	.0127898	64	71	IO1	0	0	0
19	4	PR	.2600	3100	1920	1920	.0121143	.0135891	68	1	FMEM	0	1	0
20	3	PR	.2600	3000	3456	3456	.0121143	.0123901	62	1	FMEM	0	1	0
21	15	D	.2280	1700	1700	167040	.0106233	.0127898	64	69	IO1	0	0	0
22	27	D	.1800	30	33	68360	.0083868	.0113909	57	60	IO1	0	0	0
23	20	D	.1760	1	2	330	.0082005	.0089928	45	50	IO1	0	0	1
24	9	PR	.1760	1800	1088	1088	.0082005	.0087930	44	1	FMEM	0	1	0
25	13	PR	.1500	1000	384	384	.0069890	.0061950	31	1	FMEM	0	1	0
26	12	PR	.1500	600	640	640	.0069890	.0083933	42	1	FMEM	0	1	0
27	11	PR	.1500	1000	832	832	.0069890	.0063949	32	1	FMEM	0	1	0
28	32	D	.1400	4	2	2112	.0065231	.0091926	46	50	IO1	0	0	0
29	8	PR	.1400	800	254	254	.0065231	.0061950	31	1	FMEM	0	1	0
30	10	PR	.1400	1800	1792	1792	.0065231	.0069944	35	1	FMEM	0	1	0
31	24	D	.1010	30	35	1033352	.0047059	.0051958	26	28	IO1	0	0	0
32	30	D	.0900	1	1	1320	.0041934	.0051958	26	28	IO1	0	0	0
33	33	D	.0680	1	1	4400	.0031684	.0049960	25	29	IO1	0	0	0
34	34	D	.0680	7	7	93060	.0031684	.0031974	16	18	IO1	0	0	0
35	40	D	.0010	3	3	240042	.0000466	0.0000000	0	0	IO1	0	0	0
36	28	D	.0010	20	28	26032	.0000466	0.0000000	0	0	IO1	0	0	0
37	39	D	.0010	50	66	20064	.0000466	0.0000000	0	0	IO1	0	0	0
38	36	D	.0010	100	209	30030	.0000466	0.0000000	0	0	IO1	0	0	0
39	22	DR	.0010	20	462	462	.0000466	0.0000000	0	0	IO1	0	0	0
40	42	P	.0010	1000	500	482608	.0000466	.0001998	1	1	IO1	0	0	0
41	35	D	.0001	50	245	21102	.0000047	0.0000000	0	0	IO1	0	0	0
42	38	D	.0001	100	1000	60060	.0000047	0.0000000	0	0	IO1	0	0	0
			TOTALS:	34031	32922	10069460	1.0000000	1.0000000	5004	3654		0	13	4
			ACCUMLATVE TOTALS:	34031	32922	10069480	1.0000000	1.0000000	5004	3654		0	13	4

TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 24684
TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS SCHEDULED FOR EXECUTION: 0

Figure 3-6 Initial File Assignment to IO1 of Example

file is requested (OBJect in a transaction oriented system), f) COMPLTED LOAD REQ is the number of times a record of the file is loaded, and g) CURRENT LOCTN and CURRENT STATUS indicate the current state of the file. Values of the appropriate columns correspond to the subsequent simulation model run. The mean service time for the CPU and IO devices are computed on each iteration for use in the analytical model. Also 'terminating condition 1' can be ignored since the tolerance is set to 0.0.

Each iteration in the analysis technique of the hybrid model is summarized as follows:

Iteration:	1	2	3	4
Simulation Model Results				
CPU Throughput:	209.	219.	257.	257.
f:	.33	.33	.33	.32
P1:	1.00	.93	.77	.80
P2:	0.00	.06	.20	.18
P3:	0.00	.01	.03	.02
Analytical Model Results				
CPU Throughput:	243.	271.	267.	270.
P1*:	.92	.76	.80	.81
P2*:	.06	.20	.17	.17
P3*:	.02	.04	.03	.02

The procedure terminates on iteration 4 because no new file assignment is made. The terminating file assignment is shown in Figures 3-7a, 3-7b, and 3-7c. A plot of the CPU throughput for each iteration is shown in Figure 3-8. For each iteration, the top line represents the steady-state CPU throughput obtained from the analytical model. The lower line represents the CPU throughput of the simulation model as it approaches steady-state since it is a function of the number of jobs processed.

III. OBJECT STATISTICS

A. LOGICAL MEMORY: (1. 1) MEMORY NAME: (FMEM .IO1) MEMORY CAPACITY: (100000. 99000000)

N	J	ID	F	I	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ REQ	COMPLTED LOAD REQ	CURRENT LOCTN	CURRENT STATUS XEQ	LDD	LDG
1	41	D	5.0000	75	1056	3168	.2329677	.2286171	1144	1246	IO1	0	0	3
2	18	D	1.4200	304	304	516800	.0661628	.0651479	326	352	IO1	0	0	0
3	25	D	1.1300	40	46	5630400	.0526507	.0515588	258	285	IO1	0	0	0
4	29	D	.9120	30	33	600000	.0424933	.0411671	206	226	IO1	0	0	0
5	26	D	.8430	47	47	135360	.0392784	.0405675	203	221	IO1	0	0	0
6	17	D	.8200	1	2	264000	.0382067	.0405675	203	227	IO1	0	0	0
7	5	PR	.7600	2900	254	254	.0354111	.0341727	171	1	FMEM	0	1	0
8	37	D	.7500	7	7	54012	.0349452	.0379696	190	207	IO1	0	0	0
9	19	D	.4920	3	6	90024	.0229240	.0235811	118	135	IO1	0	0	0
10	14	P	.2960	10	64	21504	.0137917	.0135891	68	68	IO1	0	0	0
11	13	PR	.1500	1000	384	384	.0069890	.0037970	19	1	FMEM	0	1	0
12	8	PR	.1400	800	254	254	.0065231	.0061950	31	1	FMEM	0	1	0
13	39	D	.0010	50	66	20064	.0000466	.0001998	1	1	IO1	0	0	0
14	36	D	.0010	100	209	30030	.0000466	0.0000000	0	0	IO1	0	0	0
15	22	DR	.0010	20	462	462	.0000466	0.0000000	0	0	IO1	0	0	0
16	42	P	.0010	1000	500	482608	.0000466	0.0000000	0	0	IO1	0	0	0
17	35	D	.0001	50	245	21102	.0000047	0.0000000	0	0	IO1	0	0	0
18	38	D	.0001	100	1000	60060	.0000047	0.0000000	0	0	IO1	0	0	0
			TOTALS:	6537	4939	7930486	.5925394	.5871303	2938	2971		0	3	3
			ACCUMLATVE TOTALS:	6537	4939	7930486	.5925394	.5871303	2938	2971		0	3	3

TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 892
TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS SCHEDULED FOR EXECUTION: 0

Figure 3-7a Terminating File Assignment to IO1 of Example

8. LOGICAL MEMORY: (1, 2) MEMORY NAME: (FMEM ,I02) MEMORY CAPACITY: (100000, 99000000)

N	J	ID	F	I	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ REQ	COMPLTED LOAD REQ	CURRENT LOCTN	CURRENT STATUS XEQ	LOD	LOG
19	31	D	.5600	50	202	126000	.0260924	.0251799	126	135	I02	0	0	0
20	21	D	.4920	10	2000	113154	.0229240	.0243805	122	139	I02	0	0	0
21	23	D	.2960	27	27	216000	.0137917	.0155875	78	85	I02	0	0	0
22	16	D	.2680	10	150	24000	.0124871	.0117906	59	63	I02	0	0	0
23	15	D	.2280	1700	1700	167040	.0106233	.0113909	57	63	I02	0	0	1
24	27	D	.1800	30	33	68360	.0083868	.0095923	48	53	I02	0	0	0
25	20	D	.1760	1	2	330	.0082005	.0075939	38	39	I02	0	0	0
26	9	PR	.1760	1800	1088	1088	.0082005	.0067946	34	1	FMEM	0	1	0
27	12	PR	.1500	600	640	640	.0069890	.0077938	39	1	FMEM	0	1	0
28	11	PR	.1500	1000	832	832	.0069890	.0085931	43	1	FMEM	0	1	0
29	32	D	.1400	4	2	2112	.0065231	.0075939	38	42	I02	0	0	0
30	10	PR	.1400	1800	1792	1792	.0065231	.0073941	37	1	FMEM	0	1	0
31	24	D	.1010	30	35	1033352	.0047059	.0045963	23	26	I02	0	0	0
32	30	D	.0900	1	1	1320	.0041934	.0059952	30	32	I02	0	0	0
		TOTALS:		7063	8504	1756020	.1466299	.1542766	772	681		0	4	1
		ACCUMLATVE TOTALS:		13600	13443	9686506	.7391693	.7414069	3710	3652		0	7	4

TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 4352
TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS SCHEDULED FOR EXECUTION: 0

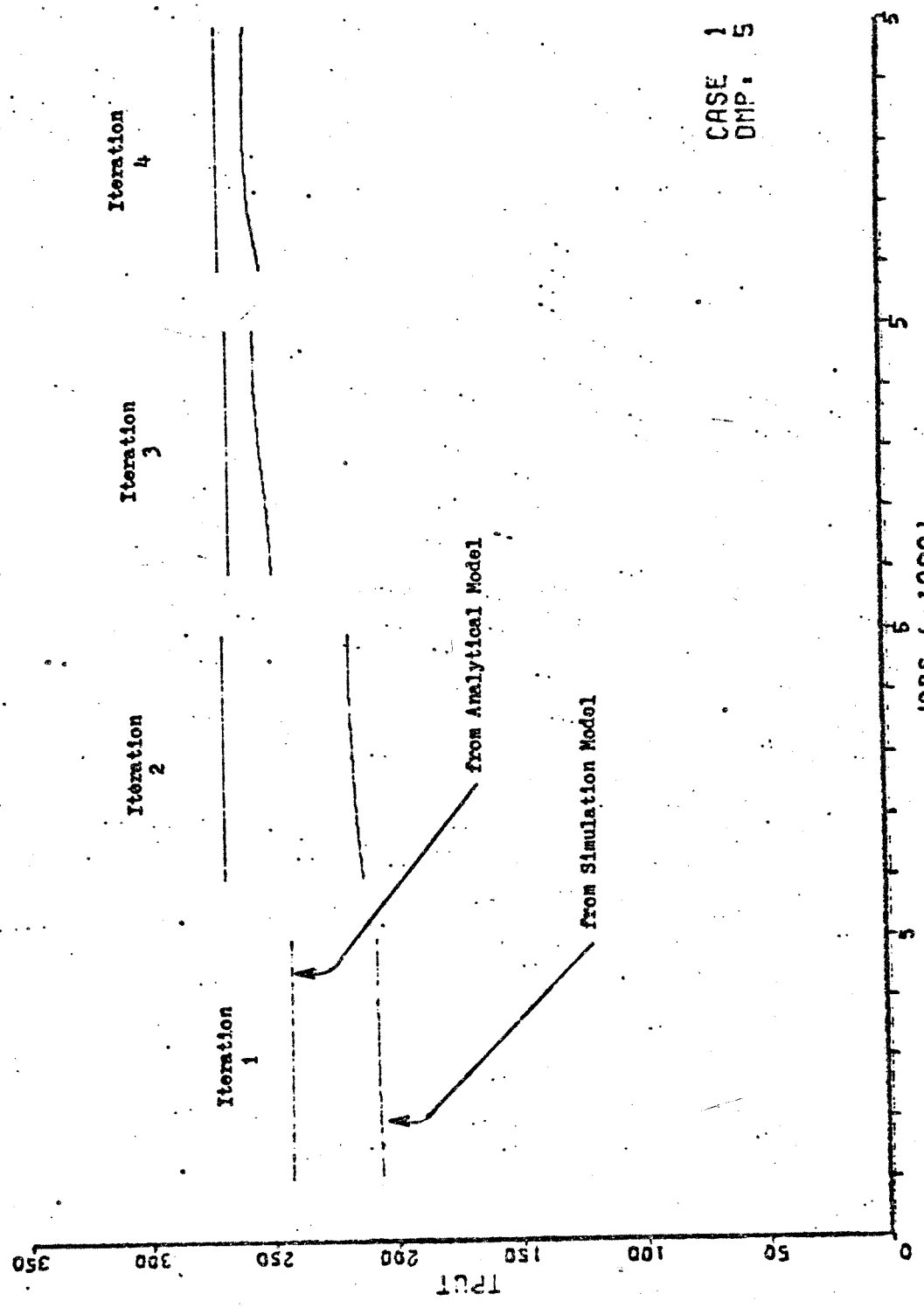
Figure 3-7b Terminating File Assignment to I02 of Example

C. LOGICAL MEMORY: (1: 3) MEMORY NAME: (FMEM :IO3) MEMORY CAPACITY: (100000: 99000000)

N	J	ID	F	I	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ REQ	COMPLTED LOAD REQ	CURRENT LOCTN	CURRENT STATUS XEQ	LOD	LOG
33	1	PR	3.1000	6000	2112	2112	.1444400	.1416867	709	1	FMEM	0	1	0
34	2	PR	1.2000	3000	6000	6000	.0559123	.0521583	261	1	FMEM	0	1	0
35	7	PR	.3200	2400	2752	2752	.0149099	.0157874	79	1	FMEM	0	1	0
36	6	PR	.3200	2900	3200	3200	.0149099	.0151878	76	1	FMEM	0	1	0
37	4	PR	.2600	3100	1920	1920	.0121143	.0139888	70	1	FMEM	0	1	0
38	3	PR	.2600	3000	3456	3456	.0121143	.0131894	66	1	FMEM	0	1	0
39	33	D	.0680	1	1	4400	.0031684	.0035971	18	21	IO3	0	0	0
40	34	D	.0680	7	7	93060	.0031684	.0029976	15	17	IO3	0	0	0
41	40	D	.0010	3	3	240042	.0000466	0.0000000	0	0	IO3	0	0	0
42	28	D	.0010	20	28	26032	.0000466	0.0000000	0	0	IO3	0	0	0
			TOTALS:	20431	19479	382974	.2600307	.2585931	1294	44		0	6	0
			ACCUMLATIVE TOTALS:	34031	32922	10069480	1.0000000	1.0000000	5004	3696		0	13	4

TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 19440
TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS SCHEDULED FOR EXECUTION: 0

Figure 3-7c Terminating File Assignment to IO3 of Example



Improvement in System Throughput of Example

Figure 3-8

3.4 Static File Assignment

Static file assignment is file assignment for a given arrival period since it assumes the use of a static activity profile (i.e., the workload is known and fixed throughout this period). The hybrid model of the analysis technique is used in experiments to isolate the cause/effect relationship of the load factors, the degree of multiprogramming, and the hardware characteristics upon optimal static file assignment. The memory management strategies are proposed for obtaining (near) optimal static file assignment using the important rate determining factors.

3.4.1 Simulation Model Verification

3.4.1.1 One Executable Memory

Before the hybrid model is used for obtaining optimal static file assignment, the simulation model of Figure 3-4 (corresponding to the system model of Figure 3-3) must be verified. A first step in verification is to make the simulation model assume conditions such that the resulting model can be analyzed analytically. The necessary conditions are: 1) sufficient executable memory to avoid memory queuing, 2) sufficient channels to avoid channel queuing, 3) known branching probabilities to the associated IO devices, and 4) known exponential service distributions for all servers (i.e., IO1, IO2, IO3, and CPU). The first two conditions are satisfied by setting the capacity of the executable memory large enough so that memory queuing cannot occur, and by setting the number of channels in the model equal to the degree of multiprogramming. The third condition is satisfied by generating three sets of non-reusable files (here, 50 files per set) and by assigning each set to an IO device after equally setting the request frequency parameter of each file to the known branching probability to that IO device divided by the number of files in the set. The fourth condition is satisfied by dividing the ordinate of the cumulative distribution of the exponential distribution (known for each server) into equally spaced points (one point per file which is assigned to that server), and then using the

corresponding value of the abscissa as the service time parameter. Of course, the appropriate parameter (i.e., instructions executed/request, or words loaded/request) is selected depending on whether the server is the CPU or an IO device. It should be noted that this method of discretely sampling a continuous distribution slightly skews the sample points toward shorter service times since the asymptote of the function is truncated. Because of this, the throughput of each server in the simulation model is slightly larger than the corresponding throughput in the analytical model. A better measure for comparison is server utilization since it is computed independently of throughput in the simulation model. Also due to transient fluctuations before steady state is reached in the simulation model, a closer comparison is expected for higher utilized servers.

After comparing the simulation model to the analytical model for a wide range of branching probabilities (for 3 above) and mean service times (for 4, above), the 'worst' case and the 'best' case (with respect to CPU utilization) parameters can given in Figure 3-9. Three observations should be made from these parameters: 1) the mean service times of the simulation model are all slightly larger than their analytical model counterparts, 2) the coefficient of variation of all simulation model servers are approximately one, and 3) the worst case model is IO bound and the best case model is CPU bound. Additional verification

	Worst Case			Best Case		
Degree of Multiprogramming	5			3		
	Analytical Model	Simulation Model		Analytical Model	Simulation Model	
Branching Probabilities						
IO1	0.4	0.4060		0.2	0.1951	
IO2	0.4	0.4008		0.4	0.3983	
IO3	0.2	0.1932		0.4	0.4066	
Mean Service Times						
CPU	0.0417	0.0386		1.0000	0.9418	
IO1	0.5000	0.4782		0.5000	0.4677	
IO2	0.3333	0.3132		0.3333	0.3163	
IO3	0.2500	0.2212		0.2500	0.2337	
Standard Deviation of Mean Service Times						
CPU	0.0417	0.0363		1.0000	0.8725	
IO1	0.5000	0.4545		0.5000	0.4362	
IO2	0.3333	0.2829		0.3333	0.2917	
IO3	0.2500	0.2068		0.2500	0.2175	
Relative Error in CPU Utilization		3.25%			0.2%	

Simulation Model Verification Results

Figure 3-9

results of the worst and best cases are given in Figures 3-10 and 3-11, respectively. The rows of Figures 3-10a and 3-11a are labelled as mean ThroughPUT, mean UTILization, mean Queue LENgth, and mean WAITing time. They contain pairs of values, the top value being generated by the simulation model and the bottom by the corresponding analytical model (obtained by using ASQ [K1]). The columns represent different degrees of multiprogramming. Figures 3-10b and 3-11b represent the service time distributions that the simulation model generated for each server in the worst and best cases. Note the similarity between these distributions and their corresponding exponential distribution.

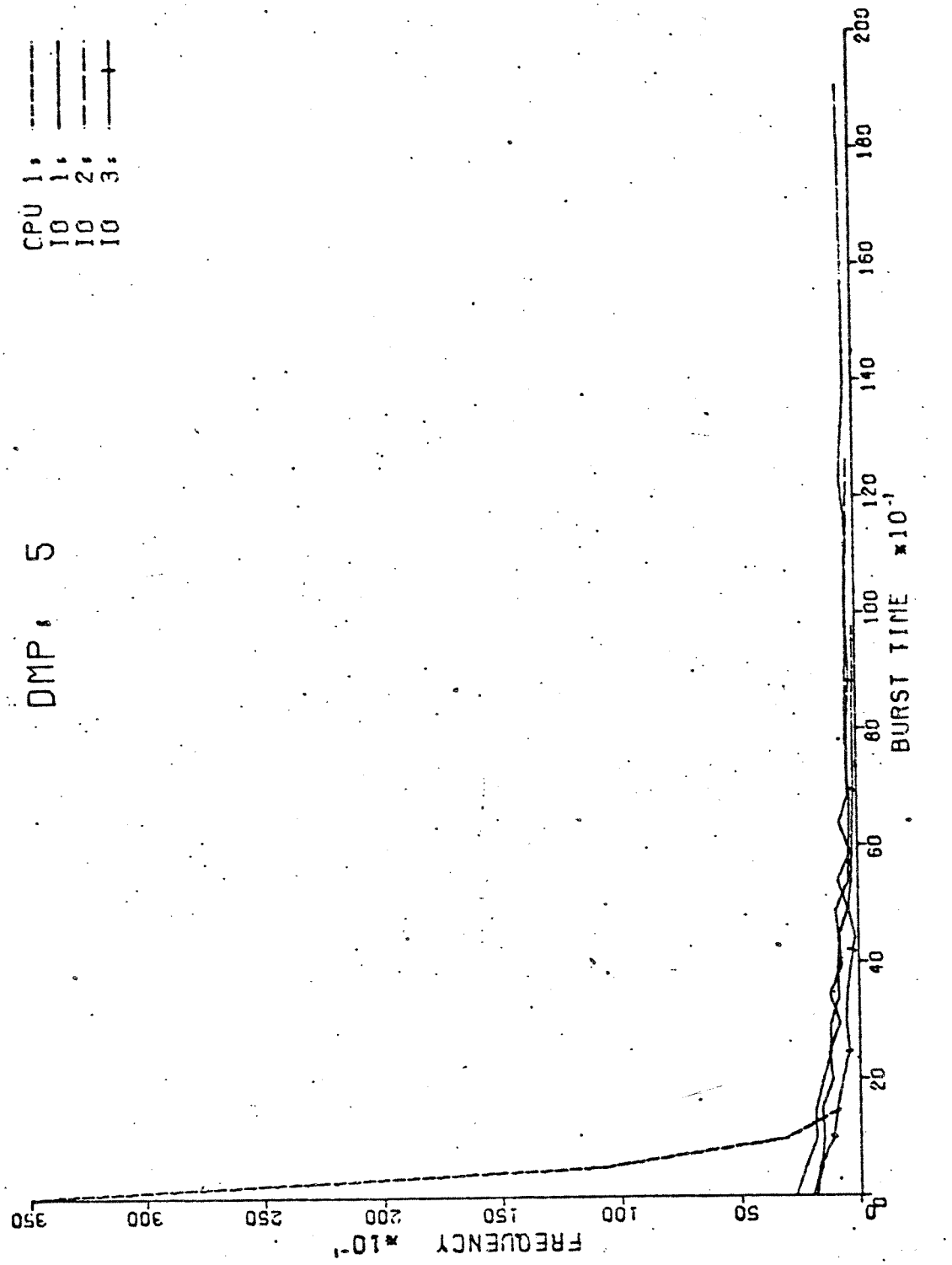
The second step in verification of the simulation model is to examine the channel queue in more detail. Since current analytical models cannot produce exact results for comparison, other simulation models must be used. Since simulation model results are compared to simulation model results, only gross differences imply potential modeling errors.

Channel utilizations are compared in Figure 3-12 using the worst case parameters of Figure 3-9. Remember that the number of channels is set equal to the degree of multiprogramming to avoid channel queuing. Each column pair, representing a fixed degree of multiprogramming, contains a left component whose values for channel utilizations generated by the simulation model of Figure 3-4 and a right component whose values are generated by an

		1	2	3	4	5	6	7	8	9	10
CPU	TPUT	2.4480 2.3530	3.6817 3.5030	4.4000 4.1130	4.8370 4.4563	4.8584 4.6586	5.0675 4.7810	5.1770 4.8580	5.2576 4.9070	5.1666 4.9390	5.0993 4.9590
CPU	UTIL	.0973 .0930	.1434 .1466	.1709 .1714	.1834 .1857	.1878 .1941	.1941 .1992	.2056 .2024	.2041 .2045	.2040 .2058	.2034 .2066
	QLEN	.0973 .0980	.1573 .1603	.1946 .1989	.2158 .2226	.2192 .2373	.2310 .2465	.2473 .2523	.2475 .2560	.2450 .2585	.2457 .2601
	WAIT	.0413 .0417	.0437 .0457	.0450 .0481	.0453 .0499	.0457 .0509	.0461 .0515	.0483 .0519	.0476 .0522	.0479 .0523	.0488 .0524
IO1	UTIL	.4668 .4706	.7056 .7036	.8322 .8226	.9068 .8912	.9438 .9316	.9569 .9562	.9730 .9715	.9812 .9814	.9866 .9877	.9981 .9919
	QLEN	.4668 .4706	1.0464 1.0300	1.4866 1.6700	2.4122 2.3400	3.2547 3.1490	4.0110 3.9670	4.7370 4.8250	5.6849 5.7170	6.5676 6.6340	7.7971 7.5720
	WAIT	.4454 .5000	.7677 .7353	.9685 1.0150	1.2265 1.3350	1.6511 1.6900	1.9704 2.0740	2.3153 2.4830	2.7473 2.9130	3.2261 3.3580	3.9047 3.8170
IO2	UTIL	.3210 .3137	.4607 .4671	.5580 .5434	.5642 .5942	.6100 .6211	.6331 .6374	.6832 .6477	.6432 .6542	.6616 .6585	.6451 .6613
	QLEN	.3210 .3137	.6067 .6136	.8863 .8450	1.1069 1.1200	1.2637 1.3170	1.4765 1.4770	1.7104 1.6040	1.6945 1.7040	1.8714 1.7800	1.6469 1.8350
	WAIT	.3325 .3333	.5215 .5379	.5005 .5379	.5874 .6233	.6524 .7067	.7334 .7722	.8260 .8256	.8214 .8680	.8992 .9012	.8162 .9268
IO3	UTIL	.1149 .1176	.1716 .1752	.2004 .2057	.2149 .2228	.2077 .2329	.2227 .2390	.2460 .2429	.2634 .2453	.2457 .2469	.2485 .2480
	QLEN	.1149 .1176	.1696 .1958	.2325 .2459	.2652 .2776	.2624 .2975	.2815 .3102	.3053 .3182	.3731 .3234	.3159 .3268	.3103 .3290
	WAIT	.2422 .2506	.2662 .2794	.2750 .2989	.2791 .3115	.2833 .3194	.2843 .3244	.2955 .3275	.3396 .3296	.3079 .3309	.2926 .3317

Worst Case Verification Results

Figure 3-10a



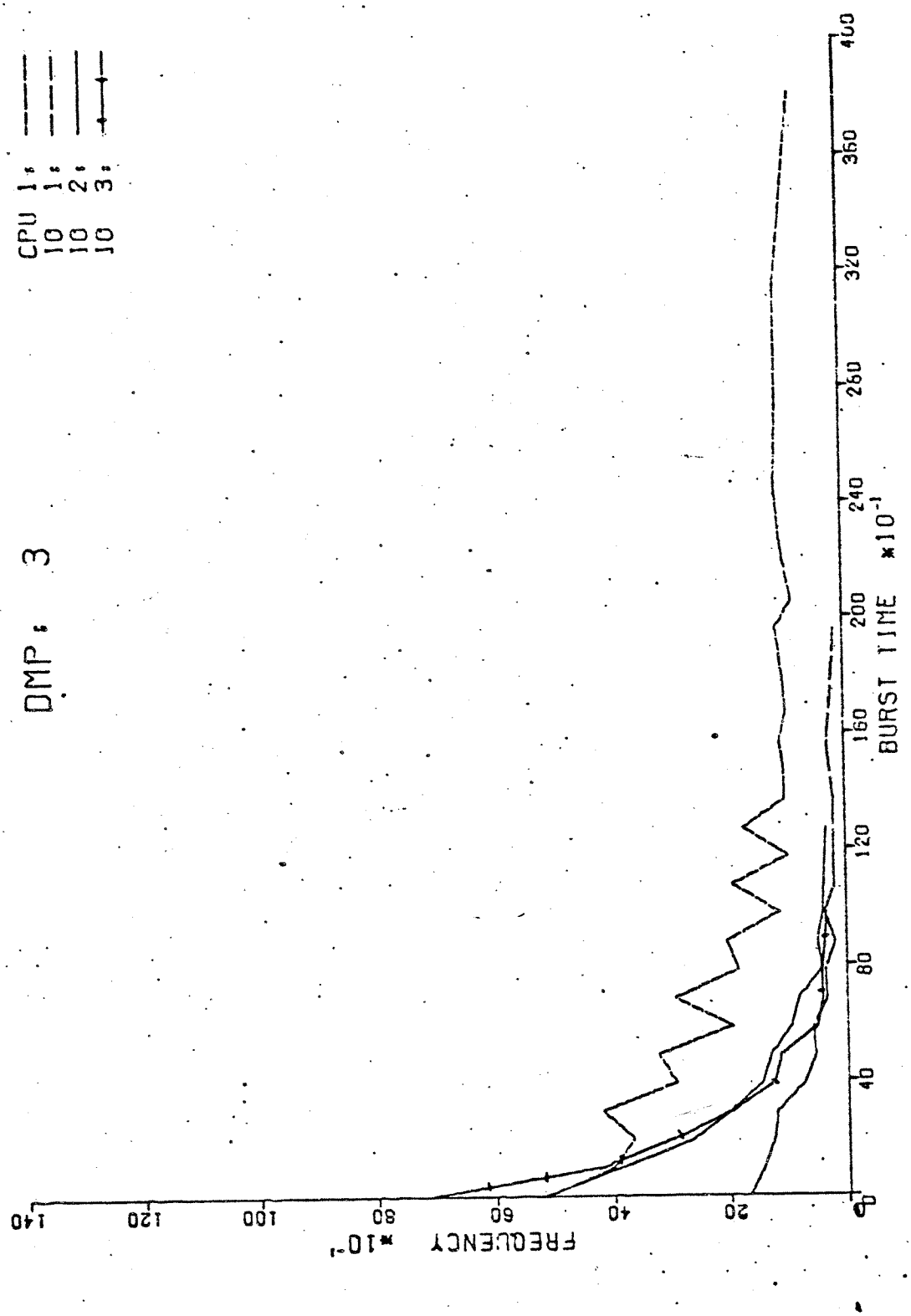
Worst Case Service Time

Figure 3-10b

		1	2	3	4	5	6	7	8	9	10
CPU	TPUT	.7796 .7500	1.0032 .9471	1.0534 .9902	1.0855 .9963	1.0873 .9997	1.0740 1.0000	1.0393 1.0000	1.0624 1.0000	1.0445 1.0000	1.0343 1.0000
CPU	UTIL	.7511 .7500	.9484 .9471	.9922 .9902	.9935 .9983	.9998 .9997	1.0000 1.0000	1.0000 1.0000	1.0000 1.0000	1.0000 1.0000	1.0000 1.0000
	QLEN	.7511 .7500	1.4579 1.4573	2.6392 2.6310	3.6266 3.6250	4.6328 4.6250	5.6331 5.6240	6.6319 6.6240	7.6210 7.6240	8.6268 8.6240	9.6305 9.6240
	WAIT	.9420 1.0000	1.4603 1.7500	2.5063 2.6570	3.3388 3.6310	4.3382 4.6250	5.2326 5.6240	6.3760 6.6240	7.1680 7.6240	8.2506 8.6240	9.3040 9.6240
IO1	UTIL	.0733 .0750	.0933 .0947	.0962 .0990	.0966 .0998	.0913 .1000	.0947 .1000	.0989 .1000	.1066 .1000	.0994 .1000	.1009 .1000
	QLEN	.0733 .0750	.0999 .1018	.1043 .1091	.1054 .1107	.0986 .1110	.1025 .1111	.1079 .1111	.1212 .1111	.1101 .1111	.1107 .1111
	WAIT	.4848 .5000	.5155 .5375	.5149 .5339	.4945 .5546	.4842 .5554	.4874 .5555	.5223 .5555	.5481 .5556	.5329 .5556	.5169 .5556
IO2	UTIL	.0990 .1000	.1241 .1263	.1327 .1320	.1350 .1331	.1384 .1333	.1358 .1333	.1304 .1333	.1323 .1333	.1334 .1333	.1354 .1333
	QLEN	.0990 .1000	.1410 .1359	.1483 .1504	.1562 .1531	.1580 .1537	.1519 .1538	.1468 .1538	.1501 .1538	.1521 .1538	.1512 .1538
	WAIT	.3234 .3333	.3524 .3467	.3590 .3796	.3609 .3835	.3717 .3844	.3630 .3846	.3627 .3846	.3669 .3846	.3737 .3846	.3779 .3846
IO3	UTIL	.0767 .0750	.0941 .0947	.1002 .0990	.1000 .0998	.1005 .1000	.1008 .1000	.1028 .1000	.0974 .1000	.1003 .1000	.0981 .1000
	QLEN	.0767 .0750	.1012 .1018	.1063 .1091	.1118 .1107	.1106 .1110	.1105 .1111	.1133 .1111	.1077 .1111	.1111 .1111	.1076 .1111
	WAIT	.2493 .2500	.2597 .2687	.2573 .2755	.2664 .2773	.2611 .2777	.2608 .2778	.2755 .2778	.2612 .2778	.2667 .2778	.2655 .2778

Best Case Verification Results

Figure 3-11a



Best Case Service Time Distributions

Figure 3-11b

1			2		3		4		5	
.9027	.9190	.9235	.9240	.9372	.9410	.9476	.9550	.9592	.9580
0.0000	0.0000	.9192	.9120	.9351	.9340	.9483	.9510	.9589	.9560
0.0000	0.0000	0.0000	0.0000	.9131	.9260	.9450	.9440	.9537	.9490
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.9433	.9420	.9562	.9480
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.9528	.9190
0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

6			7		8		9		10	
.9544	.9610	.9670	.9670	.9709	.9720	.9745	.9760	.9787	.9790
.9622	.9630	.9671	.9660	.9727	.9740	.9734	.9730	.9761	.9780
.9628	.9610	.9638	.9640	.9717	.9750	.9737	.9720	.9781	.9760
.9610	.9610	.9658	.9650	.9698	.9710	.9737	.9700	.9771	.9750
.9578	.9570	.9631	.9620	.9674	.9690	.9732	.9690	.9775	.9730
.9606	.9640	.9629	.9580	.9669	.9670	.9712	.9700	.9740	.9720
0.0000	0.0000	.9636	.9590	.9665	.9670	.9714	.9640	.9745	.9710
0.0000	0.0000	0.0000	0.0000	.9668	.9630	.9740	.9630	.9725	.9700
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.9698	.9630	.9747	.9650
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.9712	.9650

Verification Results using Multiple Channels

Figure 3-12

equivalent simulation model using the ASPOL language [C8]. Note that the top values in each column agree more closely than the bottom values since both simulation models allocate the channels on a top to bottom basis.

When the simulation models are modified so that only one channel is available for allocation (permitting channel queuing to occur) but otherwise using the same worst case parameters, the values for server utilizations are compared in Figure 3-13. The rows contain pairs of utilization values, the top value being generated by the simulation model of Figure 3-4 and the bottom by an equivalent ASPOL simulation model. Note that no gross differences occur.

The final step in verifying the simulation model is to force memory queuing and to compare the utilization of the servers. By setting the record size parameter (words loaded/request) of all files to be the same and then by setting the capacity of the executable memory large enough so that only one record at a time can be loaded, memory queuing is guaranteed. Figure 3-14 compares the utilization of the servers of equivalent simulation models for various degrees of multiprogramming. (The top value of each row is generated by the simulation model of Figure 3-4.) Note that only one channel is necessary because only one job at a time can flow past the memory queue. Also no gross differences are present.

3.4.1.2 Two Executable Memories

When two executable memories are used, more accurate

	1	2	3	4	5	6	7	8	9	10
CPU	.0972 .097	0.0000	0.0000	.1074 .110	0.0000	0.0000	.1068 .110	0.0000	0.0000	.1081 .107
IO1	.4716 .479	0.0000	0.0000	.5249 .520	0.0000	0.0000	.5321 .519	0.0000	0.0000	.5158 .507
IO2	.3161 .307	0.0000	0.0000	.3492 .352	0.0000	0.0000	.3477 .347	0.0000	0.0000	.3499 .365
IO3	.1152 .117	0.0000	0.0000	.1258 .128	0.0000	0.0000	.1202 .134	0.0000	0.0000	.1343 .127
CHL	.9028 .903	0.0000	0.0000	.9999 1.000	0.0000	0.0000	1.0000 1.000	0.0000	0.0000	1.0000 1.000

Verification Results using

a Single Channel

Figure 3-13

	1	2	3	4	5	6	7	8	9	10
CPU	.2825 .294	0.0000	0.0000	.2773 .294	0.0000	0.0000	.2779 .296	0.0000	0.0000	.2820 .299
IO1	.2839 .284	0.0000	0.0000	.2914 .280	0.0000	0.0000	.2918 .282	0.0000	0.0000	.2830 .281
IO2	.2868 .283	0.0000	0.0000	.2891 .281	0.0000	0.0000	.2892 .280	0.0000	0.0000	.2861 .280
IO3	.1417 .140	0.0000	0.0000	.1422 .144	0.0000	0.0000	.1411 .142	0.0000	0.0000	.1488 .140
CHL	.7175 .706	0.0000	0.0000	.7227 .706	0.0000	0.0000	.7221 .704	0.0000	0.0000	.7180 .701

Verification Results

for Memory Queuing

Figure 3-14

results can be obtained from the analytical model if the concept of 'different classes of customers' [B2] or (equivalently 'job typing' [C2] is used. This enables files to be typed according to the executable memory to which they are assigned. This typing information is used to generate more accurate CPU throughput since it takes advantage of the fact that files which are processed in the fast executable meory (classified as type 1 files) have a shorter mean service time than if they are processed in the slower executable memory (classified as type 2 files). Chandy et. al. [C3] have extended Buzen's computational techniques for solving analytical models when the model satisfies local balance conditions [C1]. These techniques are used since the analytical model satisfies local balance when the CPU service discipline is changed to processor sharing (PS). Refer to [B2]. It should be noted that the 'changed' model closely approximates real world conditions when the CPU discipline is PS and all IO device disciplines are FCFS. The number of jobs of each type is approximated by the integer part of the expected value for the number of files being simultaneously processed in each executable memory.

The CPU service discipline of the simulation model is also changed to PS necessitating reverification of server correctness. (PS is a commonly used queuing discipline of simulation models requiring that the queue entries be ordered on their

remaining processing time. The bookkeeping overhead for maintaining this discipline is greater than first-come-first-serve but considerably less than round-robin.) If only one executable memory is used (requiring only one job type) and the above necessary conditions are satisfied to make the simulation model analyzable analytically, the simulation model is verified against the corresponding analytical model using ASQ. After comparison is made for a wide range of branching probabilities (for necessary condition 3, above) and mean service times (for 4, above), the 'worst' and 'best' case (with respect to CPU utilization) parameters are given in Figure 3-15. Detailed verification results are given in Figures 3-16 and 3-17 for the worst and best cases, respectively. It should be noted that the PS discipline gives closer results under low CPU utilization (IO bound) or high CPU utilization (CPU bound). This is related to the number of jobs 'shared', either few or many.

By logically viewing the single executable memory of Figure 3-3 as containing several physical memories, each having a different access time and capacity, the 'PS' simulation model is easily used in experiments with more than one executable memory level. To further clarify this abstration, the resource management algorithm of the memory combination is as follows: when executable memory is requested and is available, it is allocated from the fastest memory; otherwise, the request must wait for

	Worst Case		Best Case	
Degree of Multiprogramming	7		5	
	Analytical Model	Simulation Model	Analytical Model	Simulation Model
Branching Probabilities				
IO1	0.4	0.3954	0.4	0.4064
IO2	0.2	0.2023	0.4	0.4004
IO3	0.4	0.4023	0.2	0.1932
Mean Service Times				
CPU	0.2000	0.1921	0.3333	0.3117
IO1	0.5000	0.4753	0.3333	0.3189
IO2	0.3333	0.3132	0.3333	0.3132
IO3	0.2500	0.2459	0.3333	0.2951
Standard Deviation of Mean Service Times				
CPU	0.2000	0.1768	0.3333	0.2898
IO1	0.5000	0.4312	0.3333	0.3033
IO2	0.3333	0.2853	0.3333	0.2830
IO3	0.2500	0.2296	0.3333	0.2757
Relative Error in CPU Utilization	5.64%		1.67%	

Simulation Model Verification Results

using PS CPU Discipline

Figure 3-15

		1	2	3	4	5	6	7	8	9	10
CPU	TPUT	1.8387 1.7650	2.6043 2.7270	3.6088 3.3060	4.1098 3.6760	4.3101 3.9290	4.5793 4.1070	4.6606 4.2390	4.8289 4.3360	4.7463 4.4130	4.8186 4.4740
CPU	UTIL	.3935 .3529	.5480 .5455	.6784 .6612	.7541 .7355	.8059 .7859	.8490 .8214	.8953 .8475	.9070 .8672	.9067 .8826	.9297 .8948
	QLEN	.3535 .3529	.7289 .7380	1.1294 1.1490	1.5007 1.5810	1.8623 2.0280	2.2695 2.4873	2.9666 2.9560	3.2369 3.4310	3.7235 3.9100	3.9916 4.3940
	WAIT	.1960 .2000	.2534 .2700	.3138 .3476	.3666 .4298	.4325 .5161	.4964 .6055	.6367 .6975	.6708 .7911	.7849 .8861	.8290 .9821
IO1	UTIL	.3506 .3529	.5566 .5455	.6825 .6612	.7660 .7355	.8375 .7859	.8657 .8214	.8765 .8475	.9014 .8672	.9072 .8826	.9432 .8948
	QLEN	.3506 .3529	.7485 .7380	1.1659 1.1490	1.6466 1.5810	2.2202 2.0280	2.6918 2.4870	2.8740 2.9560	3.5502 3.4310	4.1077 3.9100	4.8205 4.3940
	WAIT	.4854 .5030	.6426 .6765	.8204 .8690	.9615 1.0750	1.2731 1.2900	1.4682 1.5140	1.5643 1.7440	1.8695 1.9780	2.1950 2.2150	2.5576 2.4550
IO2	UTIL	.1151 .1176	.1799 .1818	.2195 .2204	.2435 .2452	.2457 .2620	.2685 .2738	.2954 .2825	.3225 .2891	.3011 .2942	.3133 .2983
	QLEN	.1151 .1176	.1994 .2032	.2401 .2652	.3091 .3102	.3169 .3432	.3481 .3678	.3909 .3884	.4774 .4008	.4055 .4121	.4241 .4212
	WAIT	.3230 .3333	.3557 .3725	.3750 .4011	.3829 .4217	.3859 .4367	.3882 .4477	.4108 .4559	.4733 .4621	.4311 .4669	.4232 .4707
IO3	UTIL	.1808 .1765	.2725 .2727	.3431 .3306	.3745 .3678	.4057 .3929	.4289 .4167	.4611 .4238	.4429 .4336	.4557 .4413	.4571 .4474
	QLEN	.1808 .1765	.3229 .3209	.4445 .4367	.5436 .5284	.6606 .6005	.6906 .6574	.7686 .7023	.7355 .7382	.7633 .7671	.7637 .7906
	WAIT	.2493 .2500	.2849 .2941	.3078 .3302	.3403 .3592	.3502 .3821	.3822 .4001	.4141 .4143	.3895 .4256	.4014 .4345	.4017 .4418

Worst Case Verification Results

using PS CPU Discipline

Figure 3-16

		1	2	3	4	5	6	7	8	9	10
CPU	TPUT	1.5588 1.5000	2.3898 2.2390	2.8402 2.6170	3.1064 2.8110	3.1624 2.9090	3.2153 2.9570	3.1133 2.9800	3.1899 2.9910	3.1373 2.9960	3.1059 2.9980
CPU	UTIL	.5000 .5000	.7522 .7463	.8908 .8724	.9569 .9370	.9858 .9696	.9951 .9857	.9974 .9934	.9997 .9970	.9999 .9986	.9999 .9994
	QLEN	.5000 .5000	1.1144 1.1190	1.8535 1.8490	2.6792 2.6700	3.6723 3.5580	4.5423 4.4930	5.5215 5.4560	6.5087 6.4370	7.4910 7.4270	8.5236 8.4210
	WAIT	.5000 .5000	.4711 .5600	.6535 .7065	.8433 .9497	1.1299 1.2230	1.4114 1.5190	1.7732 1.8310	2.0388 2.1520	2.3861 2.4790	2.7423 2.8090
I01	UTIL	.1980 .2000	.3052 .2985	.3579 .3490	.3862 .3748	.4094 .3878	.4057 .3943	.3906 .3973	.3973 .3988	.4007 .3995	.4067 .3998
	QLEN	.1980 .2000	.3644 .3582	.4751 .4740	.5577 .5525	.6239 .6021	.6140 .6317	.5849 .6483	.6684 .6573	.6434 .6620	.6111 .6644
	WAIT	.3234 .3333	.3824 .4000	.4750 .4527	.4479 .4913	.4931 .5175	.4820 .5340	.4808 .5439	.4918 .5494	.5240 .5524	.5070 .5540
I02	UTIL	.2044 .2000	.2990 .2985	.3602 .3490	.3816 .3748	.3966 .3878	.4017 .3943	.4108 .3973	.3962 .3988	.4017 .3995	.3930 .3998
	QLEN	.2044 .2000	.3573 .3582	.4741 .4740	.5441 .5525	.5891 .6021	.6157 .6317	.6599 .6483	.6073 .6573	.6226 .6620	.6222 .6644
	WAIT	.3325 .3333	.3930 .4000	.4169 .4527	.4510 .4913	.4680 .5175	.4853 .5340	.5329 .5439	.4881 .5494	.4956 .5524	.5087 .5540
I03	UTIL	.1976 .1000	.1481 .1493	.1728 .1745	.1841 .1874	.1803 .1939	.1885 .1971	.1973 .1987	.2132 .1994	.1990 .1997	.2019 .1999
	QLEN	.1976 .1000	.1615 .1642	.1973 .2031	.2191 .2255	.2148 .2376	.2240 .2440	.2316 .2471	.2756 .2487	.2430 .2494	.2431 .2497
	WAIT	.3230 .3333	.3495 .3667	.3616 .3881	.3591 .4010	.3571 .4085	.3612 .4125	.3731 .4147	.4142 .4157	.3912 .4162	.3774 .4165

Best Case Verification Results

using PS CPU Discipline

Figure 3-17

memory to become available. (This allows the more frequently executed files to be assigned to the faster executable memory levels.) In the experiments which follow, no more than two levels are used. This algorithm greatly simplifies simulation model construction and verification for more than one executable memory.

Verification results for the simulation model which allows two executable memories with the same access time are identical to those of the previous figures when the following parameter changes are made (for condition 1, above): the capacity of the first executable memory is set to zero, the capacity of the second executable meory is set large enough to avoid memory queuing, and all other parameters remain unchanged. The distinguishing feature is that more real time is now required to execute the simulation model. Additional tests reveal the same results when the capacity of the first executable memory is small enough so that the second executable memory must also be used to avoid memory queuing (all other parameters being the same). This indicates that the resource management algorithm for the executable memories is functioning as expected.

3.4.2 Strategies

It is of interest to evaluate the effectiveness of initial file assignment strategies. Step 'a' of the iterative procedure of the analysis technique (i.e., initially load the IO device according to some loading strategy) provides the mechanism by which different initial loading strategies (file assignments) may be evaluated. The use of 'good' strategies for this step produces closer first approximations to optimal file assignment. When weighted by the computation cost of the strategy, the 'goodness' of the initial memory management strategy can be determined.

The following initial strategies are evaluated by the above method:

(1) Load the most frequently executed files on the fastest devices using device capacity as the only loading constraint.

(2) Load the most frequently executed files on the fastest devices using device capacity and semi-optimal branching probabilities (defined below) as the loading constraints.

(3) Load the most frequently executed files after frequency normalization (i.e., request frequency divided by volume) on the fastest device using device capacity as the only loading constraint.

(4) Load the most frequently executed files after frequency normalization on the fastest devices using device capacity and semi-optimal branching probabilities as the loading constraints.

(5) Load the most frequently executed files on the slowest devices using device capacity as the only loading constraint.

The semi-optimal branching probabilities are computed via the analytical model (as described in section 3.3.2.2) when the mean IO device service time is generated by using the mean record size of all files and the device characteristics (i.e., mean size * transfer time + latency time). Strategy (5) is a worst case strategy for comparison purposes. It should be noted that in the experiments which follow the latency time of the IO devices is the dominating factor in calculating the semi-optimal branching probabilities of the initial loading strategy since the record size parameter of all files is relatively small (resulting in a small transfer time). Consequently, little difference is noticed here if the branching probabilities are calculated using latency time alone.

3.4.3 Experiments

The experiments for evaluating static file assignment strategies use the following values for the domain variables of the hybrid model. The load factors (i.e., job characteristics, workloads) are the same as those described earlier in the definitions section. The degree of multiprogramming for the model is seven; this value appears to be realistic in the sense that higher degrees of multiprogramming have little effect on system throughput (using these job and hardware characteristics). The distinguishing feature of the experiments is the difference in only one hardware characteristic: the capacity of the executable memories. The first experiment uses only one executable memory level of 32K words capacity and the second experiment uses two executable memory levels, a fast memory of 16K words capacity and a slower memory of 16K words capacity. Otherwise, the hardware characteristics are the same as those mentioned earlier.

3.4.4 Results

To lend additional credibility to the hybrid model by showing that it is functioning as expected, the optimal throughputs for the different cases are given:

	1 MEM	2 MEM
RL	140.	119.
RE	139.	115.
RP	140.	118.
NL	96.9	94.2
NE	121.	114.
NP	138.	117.

The reader should note the following: 1) all cases in which the files are usable (R*) are CPU bound yielding a throughput similar to the NP case (e.g., when the CPU is the throughput bottleneck, whether or not the files are reusable makes little difference), 2) the throughput of the NL case of both executable memory configurations are similar since executable memory access time is not the throughput bottleneck, and 3) additional executable memory is helpful only the CPU bound cases.

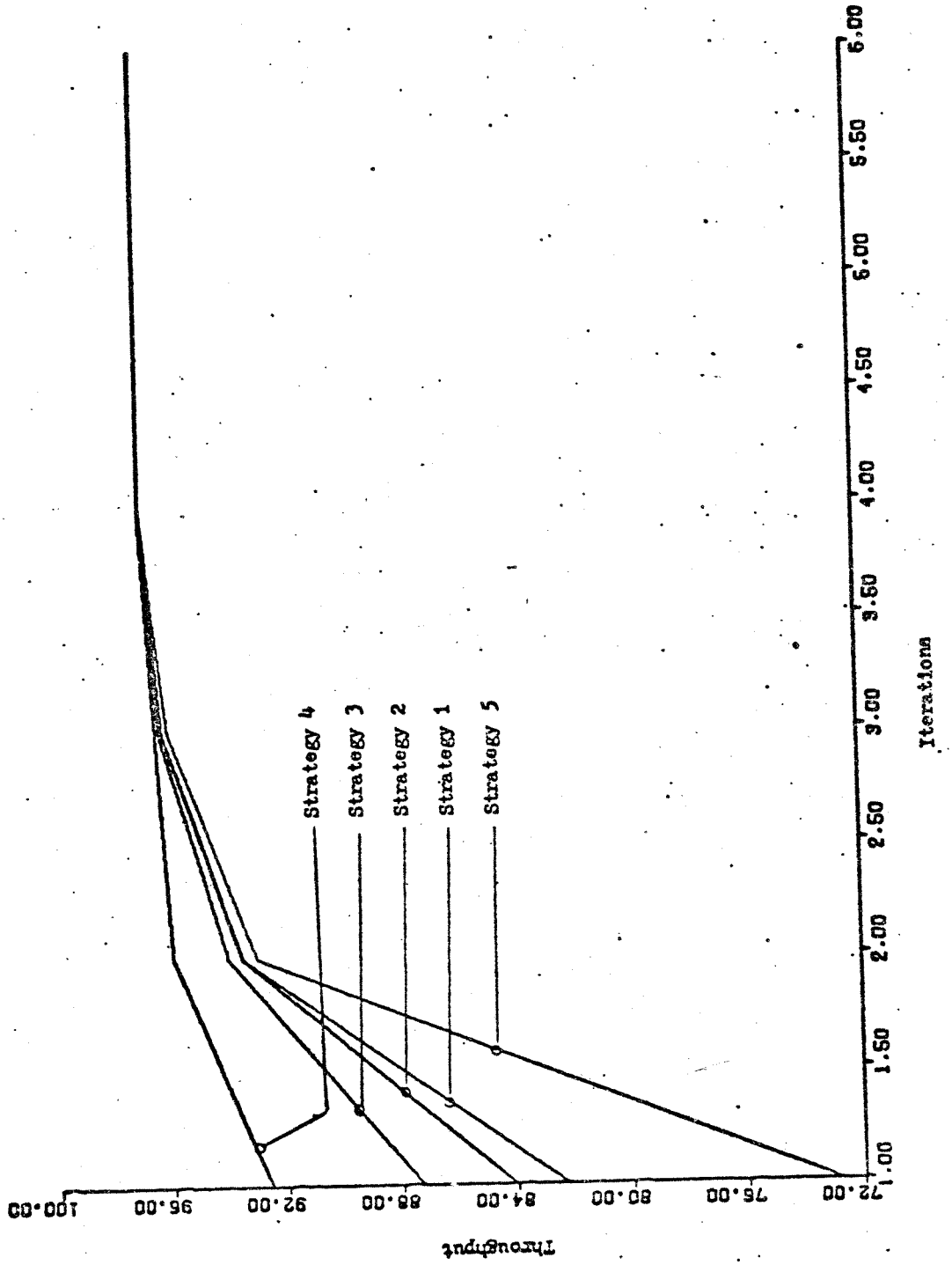
A summary of the system throughput for the various initial loading strategies and cases are now given. The values are the CPU throughputs corresponding to the first file assignment iteration. Note that the summary consists of the cases in which all files are non-reusable because all reusable file cases are almost identical to the NP case.

		Strategy				
		1	2	3	4	5
NL,	1 MEM	82.3	84.1	87.3	92.6	75.8
	2 MEM	82.0	83.5	85.2	88.7	75.6
NE,	1 MEM	109.	110.	113.	116.	105.
	2 MEM	103.	100.	107.	110.	101.
NP,	1 MEM	136.	135.	138.	137.	136.
	2 MEM	116.	115.	117.	117.	118.

The following observations from these results should be made: 1) when the system is CPU bound (case NP), there is almost no difference from the worst file assignment strategy (136. for one executable memory and 118. for two executable memories). Generalizing these results, when the system is CPU bound (regardless of job characteristics or IO device configuration), it makes little difference as to the assignment of files on the IO devices, 2) when the system is balanced (case NE), there

is a 12% difference between the worst file assignment strategy and the optimal throughput (values of 100 and 114, respectively). Since there is a 3% difference between the best file assignment strategy and the optimal throughput (values of 110 and 114, respectively), the actual improvement is 9%. With only a marginal improvement in throughput, the cost of producing the file assignment must be considered, 3) when the system is load bound (case NL), the greatest improvement in throughput can be observed. A 22% difference between the worst file assignment strategy and optimal throughput (values of 75.8 and 96.9, respectively) and a 5% difference between the best file assignment strategy (values of 92.6 and 96.9, respectively) results in a 17% improvement. (Note that the best file assignment strategy, the one which considers both the capacity and branching probability constraints, is near optimal.) This case of maximum improvement due to file assignment is discussed in more detail later, and 4) for a given strategy, the throughput difference between one executable memory and two executable memories for case NL is small because the second executable memory is rarely used, but the difference for case NP is greater because the second executable memory is relatively heavily used. Again, additional executable memory is helpful only in the CPU bound case.

Figure 3-18 gives the throughput results of the load bound case. For each strategy, throughput is a function of the number of iterations until convergence to optimal throughput is



Throughput Convergence of Different

Loading Strategies for the Load Bound Case

Figure 3-18

achieved. As has been noted earlier, strategy 5 (i.e., loading the most frequently executed files on the slowest devices using device capacity as the only loading constraint) is a worst case strategy given here for comparison purposes. The remaining four strategies represent those which conceivably might be used. The significant factors involved in these strategies are 1) frequency normalization, and 2) utilizing the branching probabilities as a loading constraint. The only difference between strategies 1 and 2 and strategies 3 and 4 is frequency normalization (i.e., request frequency divided by volume). The result is that the later strategies have better initial throughput than the former. Also a secondary effect is noted. When the frequency is normalized, the capacity constraint does not dominant the branching probability constraint as quickly. In other words, files can be loaded on an IO device such that the branching probability is better satisfied before the capacity of the device prevents additional loading. The initial approximations of strategies 1 and 2 are closer together than the initial approximations of strategies 3 and 4. The only difference between strategy 3 and strategy 4 is the utilization of the branching probabilities as a loading constraint. The result indicates an improvement of 6.1% in throughput when this constraint is not ignored. Since the time cost of computing the semi-optimal branching probabilities is quite reasonable (less than 20 CPU seconds on the CDC 6600), near-optimal loading strategies must include this constraint. In summary, the throughput improvement of strategy

4 over strategy 5 is over 20% emphasizing that file assignment is a crucial factor in system throughput when the system is load bound.

CHAPTER IV

THE UT2D PERIPHERAL PROCESSOR LIBRARY -- A CASE STUDY

4.1 Introduction

The UT2D operating system is a system which coordinates the activities of a CDC 6600 and a CDC 6400. Essentially, it is a pair of autonomous operating systems which communicate to share resources such as mass storage (e.g., extended core storage (ECS), disks), permanent files, and certain system libraries (e.g., Peripheral Processor Library). Normally, the 6600 system handles batch jobs and the 6400 system handles interactive jobs. Since batch jobs produce a greater variety of resource demands on the system, trace data from the 6600 is used to parameterize this case study.

The CDC 6600 computer system is composed of 10 small processors called peripheral processing units (PPUs) in addition to the central processor. The purpose of these PPUs is to perform input/output and control functions in support of the central processor. All PPUs have access to 12 channels which are in turn connected to various IO devices (i.e., memories). Data transfer on the channels is controlled by instructions issued by the PPUs and can provide either single word or block transfer from the devices. Each PPU has its own memory of 4096 12-bit word capacity which is separate from the 6600's central memory. The peripheral processors act as a buffer between the external environment and the central processor.

An important function of the operating system is to coordinate the activity of the various PPUs. Communication between the operating system and the PPUs is accomplished through communication areas (i.e., mailboxes) in central memory. For example, a PPU 'idles' in its resident program by checking that word 0 of its communication area remains cleared. Whenever the operating system wishes a PPU to perform some function (such as transferring data between central memory and a disk unit), it enters the appropriate function name into word 0 of the allocated PPU's communication area. After the resident program 'senses' that word 0 is no longer cleared, it must then locate the requested transient program in the Peripheral Processor Library. (This library may reside in many storage levels of a memory (IO device) hierarchy.) After this program is located, it is loaded into the PPU's memory and executed. Following completion of the transient program, word 0 is cleared and the PPU idles back in its resident program. The (pseudo) IO devices from which the PPU loads this transient program are central memory, ECS, and the system disk. Additional information concerning the operation of the UT2D operating system and the CDC 6600 hardware system can be obtained in [H2, T2].

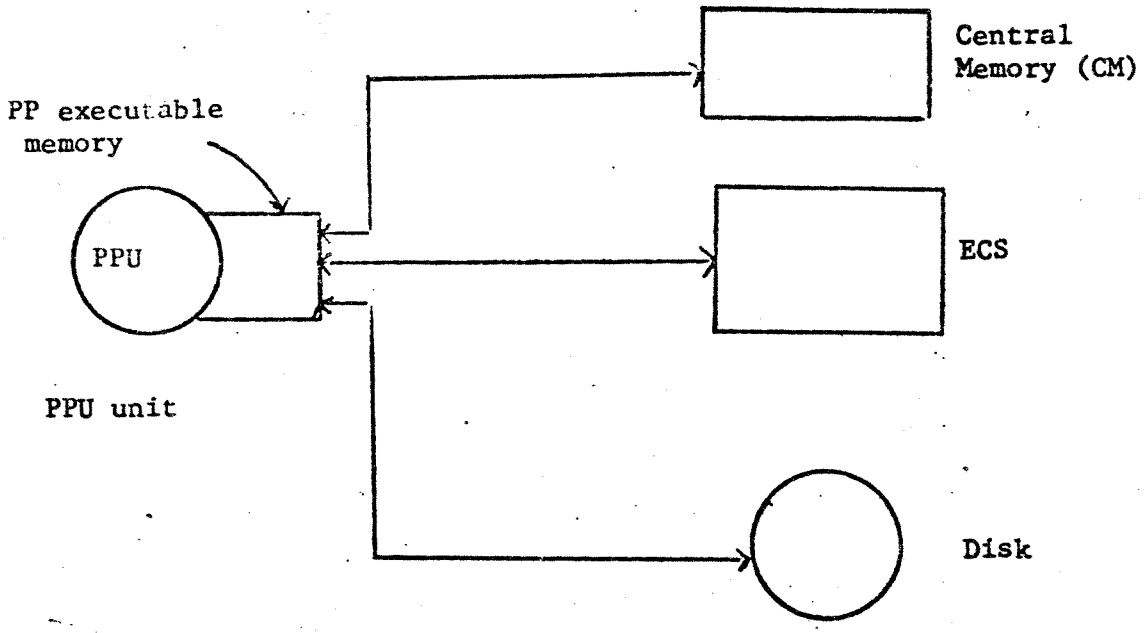
It is the purpose of this case study to indicate where to assign the programs of the Peripheral Processor Library in the memory hierarchy so as to maximize the throughput of the PPU subsystem. This is accomplished by varying the capacity constraints of the three IO devices in order to produce optimal system throughput as a function

of device capacity. In this manner, a near-optimal capacity solution to this assignment is obtained.

4.2 The Model

The model interconnection topology is given in Figure 4-1. The system consists of four servers, central memory, ECS, a disk unit, and a PPU. The simulation queuing model corresponding to the system model is given in Figure 4-2. It is interpreted in the following way. First, a request for a program in the Peripheral Processor Library must queue for the secondary memory in which the program is loaded. The request is then serviced implying the transfer of the program into the executable memory of a PPU. Upon completion of the loading process, the program is executed by the PPU. After completing PPU service, the request recirculates in the model becoming a new request. The total number of program requests circulating in the model (the degree of multiprogramming for the model) is the same as the number of available PPUs. Consequently, after a program is loaded, no queuing for a PPU is required. Also note that no explicit inclusion of executable memory is necessary since each PPU and its executable memory can be viewed as a unit. The PPU server in the model stands for a set of PPUs (and associated executable memories) equal to the degree of multiprogramming (i.e., one PPU per program).

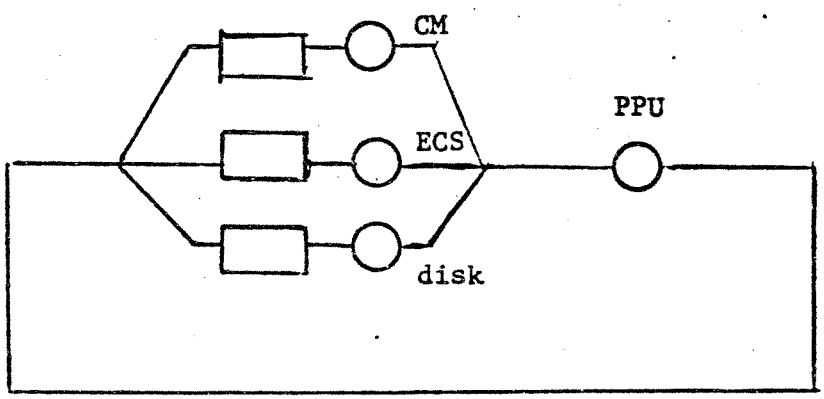
The same hybrid model that is used in the experiments with one executable memory can be used here. All that is required is a modification of a few hardware characteristics (see III.2.4) of the simulation model of Figure 3-4. They are as follows (the queue and



PPU

System Model

Figure 4-1



PPU

Simulation Model

Figure 4-2

server names are the same as those of Figure 3-4):

1) an infinite amount of executable memory for allocation in the 'Mem' queue,
2) an infinite number of channels for allocation in the 'Chl' queue,
3) setting the number of 'CPUs' available equal to the degree of multiprogramming.

The first two modifications eliminate the possibility of queuing for executable memory or channels. The third modification also eliminates queuing but also allows PPU holding while the program is being executed. All else remains the same; no simulation program changes are necessary.

4.3 Model Parameterization

The following parameters are assumed to accurately and sufficiently characterize the behavior of the CDC 6600 PPU subsystem (i.e., the loading and executing of transient programs from the PPU Library) using the UT2D operating system. Included in the parameters themselves are the effects of inter-machine interference (on shared resources such as the PPU Library) since both the CDC 6400 and the CDC 6600 were operational when the event recorder was gathering data on the 6600.

4.3.1 Activity Profile

As before, the activity profile is composed of five parameters for each program in the PPU Library: reusability of the program, request frequency of the program, instructions executed/request, words loaded (record size)/request, and volume of the program. The activity profile is given in Figure 4-3. (Note that the record size and the volume parameters are given in octal for convenience. This is the only figure in which octal notation is used.) The parameters given 'by definition' are reusability, record size, and volume. All reusability parameters are 'P' since each file is a program which is not serially reusable (i.e., a new copy must be loaded upon each request). The corresponding record size and volume parameters are equal since the entire program is loaded upon request.

The parameters that are more sensitive to system behavior are request frequency and instructions executed/request. These are obtained from a summary of the trace data (event sequences) generated by the event recorder. A detailed description of this summary is given in [H3]. The request frequency parameter is simply a count of the number of times that a given PPU transient program is located (refer to section 4.1). The instructions executed/request parameter is obtained from the mean time between consecutive PPU transient program locations. However, some of the transient programs reside in central memory while others reside on the system disk. If a program resides on the disk, the mean time must be adjusted to remove the effects of queuing for the system channel (and, at the same time, the system

Name	Reusability	Frequency	Instructions	Record Size	Volume
2WD	P	11861	93	125	125
2RD	P	11289	95	122	122
2E1	P	41	25651	137	137
1RJ	P	586	941	346	346
2MT	P	3432	119	712	712
1SJ	P	567	680	265	265
1DB	P	3853	67	231	231
LDR	P	578	259	307	307
2PD	P	3367	46	263	263
RFL	P	1396	69	170	170
2TS	P	910	100	536	536
CIO	P	22960	3	174	174
1SS	P	392	174	432	432
1PL	P	585	63	224	224
1TD	P	148	342	164	164
CPU	P	3983	9	416	416
2PU	P	352	49	412	412
PFM	P	1214	28	460	460
2WM	P	2102	16	277	277
EPR	P	1196	16	565	565
3AJ	P	165	54	145	145
2SP	P	140	60	520	520
1CJ	P	124	56	73	73
3EA	P	60	171	264	264
PCC	P	174	14	55	55
2TJ	P	116	42	32	32
2FE	P	117	32	12	12
OPE	P	1338	7	110	110
2AM	P	97	36	76	76
2JE	P	109	21	17	17
1PS	P	2208	3	212	212
2E2	P	46	50	140	140
SNP	P	760	6	67	67
2DF	P	875	4	37	37
1AJ	P	900	4	246	246
RCC	P	48	16	47	47
1SR	P	28	32	204	204
MSG	P	745	3	46	46
CLO	P	28	17	14	14

Activity Profile
of PPU Transient Programs

Figure 4-3

disk), disk seek delay, disk rotation delay, and disk transfer delay. This adjustment is more easily understood by referring to the following event sequence (which is normally generated when a PPU transient program is loaded and executed):

from CM:	from the System Disk:
1. locate program x	1. locate program a
	2. request system channel
	3. allocate system channel
	4. release system channel
5. locate program y	5. locate program b

The event trace summary contains the mean time between events 1 and 5 (for both cases). It also contains the mean time between events 2 and 3 (here, 35 ms) corresponding to the software queuing for the system channel. Between events 3 and 4, the PPU transient program is loaded from the system disk. Using the hardware characteristics discussed later, a mean load time is 53.4 ms (mean seek time is 25 ms, mean rotation time is 26 ms, and mean transfer time is 2.4 ms). Since events 2 through 4 are unique to loading from the system disk and their mean times are known, the mean time between events 1 and 5 can now be adjusted accordingly. Also, if a program's request frequency is less than 25, it is discarded as never having been loaded because the time between events 1 and 5 probably does not characterize its mean time sufficiently well.

4.3.2 Degree of Multiprogramming

Another major parameter of the model is the degree of multiprogramming. It is the parameter which specifies the amount of potential queuing interference due to requests for PPU transient programs which reside on the same IO device. The event trace summary contains the mean number of PPUs allocated for the trace interval. Its value is 3.89. Since a PPU can only be executing a single transient program at any given time, the degree of multiprogramming is set to four.

4.3.3 System Model -- Hardware Characteristics

The hardware characteristics assumed by the model are given below. All times are given in microseconds; the transfer times are given in units of either (60 bit) words or (64 X 60 bit) physical record units (PRUs). The capacity constraints are variable as stated in the purpose of this case study. The parameters are defined as follows:

A. Four PPUs with a mean execution time/instruction of 1000

B. Three IO Devices

1. Central Memory (CM)
    a. Capacity of 0, 2000, 4000, and 6000 words
    b. Mean latency time of 2000
    c. Transfer time/word of 5
2. Extended Core Storage (ECS)
    a. Capacity of 0, 2000, 4000, and 6000 words
    b. Mean latency time of 6000
    c. Transfer time/PRU of 2000
3. (CDC 808) System Disk
    a. Capacity of infinity
    b. Mean latency time of 51000
        Mean seek time of 25000
        Mean rotation time of 26000
    c. Transfer time/PRU of 1000

Note the following comments about these parameters:

1) The mean execution time for a PPU transient program is obtained by multiplying mean number of instructions executed per request (a parameter in the activity profile) by the execution time/instruction. Since the mean number of instructions executed is computed using milliseconds, the execution time/instruction is also a millisecond.
2) Since the total volume of all observed PPU transient programs is approximately 6000 words, the capacity constraints for CM and ECS vary in steps of 2000 words. The capacity of the system disk is arbitrary to allow programs not loaded in CM or ECS to be assigned to it. Its capacity constraint is set to infinity.
3) Since CM and ECS have no seek and rotational delays, the latency time parameter corresponds to the overhead associated with transferring programs from these devices.
4) Brice [B6] has shown that the mean seek time for the disks varies from 19 ms to 31 ms depending primarily on the workload. A mean seek time of 25 ms is used here.
5) The transfer time/PRU is twice as long for ECS than for the system disk since the PRU must intermediately flow through CM.

Additional information concerning hardware characteristics can be obtained from the following references [B6, C7, T2]. Professor John H. Howard, Jr. supplied parameter definitions using ECS since program loading from this device is not currently implemented.

4.4 Model Validation

The purpose of validation is to establish the credibility of the model by comparing its results with known results obtained from the actual system. It is an indication of how well the model itself reflects the actual system. If poor validation is observed, the input parameters as well as the level of detail included in the model are questioned.

By setting the capacity of CM to 2000 words, ECS to 0 words, and the system disk to infinity, and holding all other parameters (activity profile, degree of multiprogramming, and the system model) the same as those given in the previous section, the throughput as computed by the model has a value of 59. The observed throughput of the actual PPU subsystem is 52. The model produces a higher value for throughput because (1) a constant degree of multiprogramming of 4 could not be sustained by the actual system (i.e., the degree of multiprogramming sometimes well below 4), and (2) the capacity of CM is slightly larger than the corresponding capacity in the actual system. However, it is felt that these two values compare sufficiently well to establish the credibility of the model.

4.5 Model Results

It should be observed that generation of optimal throughput of the PPU subsystem does not necessarily optimize throughput of the entire computer system. For example, optimal PPU subsystem throughput may use many more words of CM for transient program assignment than another near-optimal assignment. The result is that while PPU subsystem is optimized, throughput of the computer system is degraded due to a decrease in the degree of multiprogramming caused by fewer CM words available for user programs. This observation must be remembered when evaluating the following results.

4.5.1 Throughput and Assigned Memory

The table in Figure 4-4 gives optimal throughput and the associated assigned memory of the IO devices (i.e., how much capacity of each device is actually used) for various memory capacity constraints. System disk assignment is computed by subtracting the assigned memory values for CM and ECS from the total transient program volume (i.e., 5819 words).

The entire table is not complete because a) some constraint combinations are deemed impractical (such as 0 CM and 0 ECS), and b) some entries can be implied from other entries (such as 2000 CM, 6000 ECS can be implied from 2000 CM, 4000 ECS).

The following important observations can be made from Figure 4-4. First, by increasing the capacity of CM from 0 words to 2000 words, the corresponding increase in throughput is approximately 9% in all cases. Increasing the capacity of CM beyond 2000 words does not affect throughput in any case. Second, by increasing the capacity of ECS from 0 words, the corresponding increase in throughput is near zero for all cases. So it would appear that when considering the PPU subsystem alone, the appropriate capacities for CM and ECS are 2000 and 0, respectively. However, it is noted that the relative difference in throughput between the 0 CM, 2000 ECS and the 2000 CM, 0 ECS combinations is approximately 6.5%. Since this difference is so small, the former combination is desirable over the later combination in terms of both storage costs and optimizing the entire computer system's

CM \ ECS	0	2000	4000	6000
0	NE	55 0 1996	55 0 3992	55 0 5819
2000	59 1996 0	60 1996 1969	60 1996 3823	NE
4000	60 3992 0	60 3992 1827	NE	NE
6000	60 5819 0	NE	NE	NE

Legend

A. Margins -- IO device capacity constraints

B. Entries

1. NE -- Not Evaluated

2. Values:

a. Throughput of the PPU Subsystem

b. Actual CM assigned

c. Actual ECS assigned

Results of
PPU Transient Program Assignment

Figure 4-4

performance (since an additional 2000 CM words are available for user programs at a cost of 2000 ECS words). This analysis indicates that the PPU library which is currently stored in CM should be transferred to ECS, thus freeing CM for other uses.

4.5.2 Program Assignment

Assignment of the programs in the activity profile of Figure 4-3 to be IO devices is given in Figures 4-5a and 4-5b. This assignment corresponds to the table entry of 0 CM, 2000 ECS which generates a throughput of 55. Note that less frequently requested programs are often unexpectedly assigned to faster IO devices to more closely match the optimal branching probability of that device with the total frequency requests of the programs assigned to that device.

0: LOGICAL MEMORY: (1: 2) MEMORY NAME: (FMEM ,ECS) MEMORY CAPACITY: (1000000, 20000)

N	I	ID	P	V	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ REQ	COMPLETED LOAD REQ	CURRENT LOCTN	CURRENT STATUS REQ	LOD	LOG
1	CIO	P	22960.0	3	124	124	.2910382	.2948231	1475	1475	FMEM	2	2	0
2	2WD	P	11861.0	93	85	85	.1503486	.1511093	756	756	ECS	0	0	1
3	2RD	P	11249.0	95	82	82	.1430990	.1341195	671	671	ECS	0	0	0
4	CPU	P	3743.0	9	270	270	.0504880	.0521687	261	261	ECS	0	0	0
5	1DD	P	3853.0	67	153	153	.0488402	.0517689	259	259	ECS	0	0	0
6	2MT	P	3432.0	119	458	458	.0434036	.0449730	225	225	ECS	0	0	0
7	2RD	P	3347.0	46	179	179	.0426797	.0409754	205	205	ECS	0	0	0
8	1PS	P	2208.0	3	138	138	.0279883	.0285929	143	143	ECS	0	0	0
9	2WM	P	2102.0	16	191	191	.0266447	.0275834	138	138	ECS	0	0	0
10	REL	P	1396.0	69	120	120	.0176956	.0195892	98	98	ECS	0	0	0
11	OPE	P	1338.0	7	72	72	.0169603	.0183890	92	92	ECS	0	0	0
12	2DF	P	875.0	4	31	31	.0110914	.0105936	53	53	ECS	0	0	0
13	SKP	P	760.0	6	55	55	.0096337	.0123926	62	62	ECS	0	0	0
14	MSG	P	745.0	3	38	38	.0094435	.0075954	38	38	ECS	0	0	0
			TOTALS:	540	1996	1996	.8894537	.8946632	4476	4475		2	2	1
			CUMULATIVE TOTALS:	540	1996	1996	.8894537	.8946632	4476	4475		2	2	1

TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 249
TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS SCHEDULED FOR EXECUTION: 248

PPU Transient Program Assignment to ECS

Figure 4-5a

C. LOGICAL MEMORY: (1, 3) MEMORY NAME: (FMEM ,DISK) , MEMORY CAPACITY: (1000000. 9999)

N	J	In	F	τ	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ RFQ	COMPLTED LOAD REQ	CURRENT LOCTN	CURRENT STATUS XEQ	LDD	LDG
15	PFM	P	1214.0	28	304	304	.0153885	.0155906	78	78	DISK	0	0	0
16	EPR	P	1196.0	16	373	373	.0151609	.0167899	84	84	DISK	0	0	0
17	2TS	P	910.0	100	350	350	.0115350	.0131921	66	66	DISK	0	0	0
18	1AJ	P	900.0	4	166	166	.0114083	.0109934	55	55	DISK	0	0	0
19	1PJ	P	596.0	241	230	230	.0074281	.0055966	28	28	DISK	0	0	0
20	1PL	P	585.0	63	148	148	.0074184	.0063962	32	32	DISK	0	0	0
21	LCR	P	578.0	259	199	199	.0073267	.0057965	29	29	DISK	0	0	0
22	1SJ	P	567.0	180	181	181	.0071872	.0059964	30	30	DISK	0	0	0
23	1SS	P	392.0	174	282	282	.0049689	.0035978	18	18	DISK	0	0	0
24	2PU	P	352.0	49	266	266	.0044619	.0035978	18	18	DISK	0	0	0
25	PCC	P	174.0	14	45	45	.0022056	.0027983	14	14	DISK	0	0	0
26	3AJ	P	165.0	54	101	101	.0020915	.0013992	7	7	DISK	0	0	0
27	1TD	P	148.0	142	116	116	.0018760	.0019988	10	10	DISK	0	0	0
28	2SP	P	140.0	60	336	336	.0017746	.0029982	15	15	DISK	0	0	0
29	1CJ	P	124.0	56	59	59	.0015718	.0013992	7	7	DISK	0	0	0
30	2FE	P	117.0	32	10	10	.0014831	.0005996	3	3	DISK	0	0	0
31	2TJ	P	116.0	42	26	26	.0014704	.0011993	6	6	DISK	0	0	0
32	2IE	P	109.0	21	15	15	.0013817	.0013992	7	7	DISK	0	0	0
33	2AM	P	97.0	36	62	62	.0012296	.0003998	2	2	DISK	0	0	0
34	3CA	P	60.0	171	180	180	.0007606	.0003998	2	2	DISK	0	0	0
35	HCC	P	48.0	16	39	39	.0006084	.0009994	5	5	DISK	0	0	0
36	2F2	P	46.0	50	96	96	.0005831	.0005996	3	3	DISK	0	0	0
37	2F1	P	41.0	25651	95	95	.0005197	.0003998	2	2	DISK	0	0	0
38	CIO	P	28.0	17	12	12	.0003549	.0005996	3	3	DISK	0	0	0
39	1SR	P	28.0	32	132	132	.0003549	.0005996	3	3	DISK	0	0	0
			TOTALS:	28208	3623	3823	.1103463	.1053368	527	527		0	0	0
			ACCUMLATVE TOTALS:	29448	5819	5819	1.0000000	1.0000000	5003	5002		2	2	1

TOT.L RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 0
TOT.L RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS SCHEDULED FOR EXECUTION: 0

PPU Transient Program Assignment to Disk

Figure 4-5b

CHAPTER V

CONCLUSIONS

5.1 Summary of Results

The major contributions of this work are listed as follows:

1. An analysis technique has been implemented for the automatic generation of optimal system throughput and corresponding file assignment which includes queuing delays.
2. This analysis/evaluation tool, the hybrid model, can be used as a primitive in a higher level, automatic design process.
3. The optimal branching probabilities to the IO devices are often shown to have a significant influence on the system throughput. These probabilities as well as device capacities have been identified as important constraints for practical loading strategies.
4. The hybrid model can be used to produce results which can be compared to those of more sophisticated (future) analytical models (for verification).

Two file systems, a transaction oriented system and an operating system library, have been investigated using the hybrid model and results have shown to be applicable to real-world problems. For example, given a set of files with known characteristics and a

set of secondary storage units with known characteristics, results are produced which assign particular files to particular devices such that system throughput is optimized.

It should be noted that the overhead cost of maintaining optimal file assignment is best justified in the load (IO) bound cases.

It has been illustrated that the application of this analysis technique is not limited to systems consisting of a single CPU and a single executable memory, and has been shown to be practical with respect to the time and cost needed to produce the results.

A final important point is that the hierarchical approach used in structuring the file assignment problem produces clarity in the quantitative definition of the problem.

5.2 Extensions

The analysis technique presented here uses many simplifying assumptions about the input parameters as well as the hybrid model. Future work includes relaxing some of these assumptions. For example, it is assumed that file (job) characteristics are sufficiently characterized by five parameters (i.e., reusability, frequency of file request, processing time per request, loading time per request, and file volume). A more correct characterization would include the fact that files are not requested independently of one another. This might be specified by including a sixth parameter giving a list of next files to be processed along with the corresponding probabilities of doing so. Another example involves the ability to analyze more exact analytical models that include resource holding (such as holding a channel in addition to a device for data transfer).

When the job characteristics change from one period to the next period (as with a dynamic activity profile), an algorithm should be designed to balance the performance tradeoff between allowing a file to be loaded from a slower device and percolating (inducing overhead) that file to a fast device and then loading it.

Finally, the long range goal involving this analysis technique is to generate a completely automatic design process, one in which the input parameters might be job characteristics, hardware characteristics (such as timings and costs), a total budget, and performance constraints. The output would be a system designed for

the job characteristics which would satisfy budget and performance constraints.

BIBLIOGRAPHY

[A1] Abato, J., and Durner, H. Optimizing the performance of a drum-like storage. IEEE Trans. C-18, 11 (November 1969), 992-997.

[A2] Anacker, W., and Wang, C. P. Performance evaluation of computing systems with memory hierarchies. IEEE Trans. EC-16, 6 (December 1967), 764-772.

[A3] Arora, S., and Gallo, A. Optimal sizing, loading and reloading in a multi-level memory hierarchy system. Proc. AFIPS 1971 SJCC, V38, 337-344.

[A4] Arora, S., and Gallo, A. Optimization of static loading of multilevel memory systems. JACM 20, 2 (April 1973), 307-319.

[B1] Baskett, F., Browne, J. C., and Raike, W. M. The management of a multi-level non-paged memory system. Proc. AFIPS 1970 SJCC, V36, 459-465.

[B2] Baskett, F., Chandy, K. M., Muntz, R. R., and Palacios-Gomez, F. Open, closed, and mixed networks of queues with different classes of customers. JACM, to appear.

[B3] Belady, L. A. A study of replacement algorithms for virtual storage computers. IBM Sys. J. 5, 2 (1966), 78-101.

[B4] Belady, L. A., and Kuehner, C. J. Dynamic space sharing in computer systems. CACM 12, 5 (May 1969), 282-288.

[B5] Belady, L. A., Nelson, R. A., and Shedler, G. S. An anomaly in the space-time characteristics of certain programs running in paging machines. CACM 12, 6 (June 1969), 349-353.

[B6] Brice, Richard S. A study of feedback coupled resource allocation policies in a multiprocessing computer environment, The University of Texas at Austin Computation Center and Department of Computer Sciences TSN-35, (Ph.D. Dissertation), The University of Texas at Austin, 1973.

[B7] Buzen, J. Analysis of system bottlenecks using a queuing network model. Proc. ACM Workshop on System Performance and Evaluation, Harvard Univ. Press, Cambridge, Mass., (April 1971), 82-103.

[C1] Chandy, K. M. The analysis and solutions for general queueing networks. Proc. Sixth Annual Princeton Conf. on Information Sciences and Systems, Princeton Univ., Princeton, N. J., (March 1972), 224-228.

[C2] Chandy, K. M., Keller, T. W., and Browne, J. C. Design automation and queueing networks: an interactive system for the evaluation of computer queueing models. Proc. Ninth Annual Design Automation Conference 9, (June 1972), 357-367.

[C3] Chandy, M., Herzog, U., and Woo, L. Parametric analysis of queuing network models. IBM J. Res. Develop., to appear. (Also: IBM Res. Rep. RC 4730, IBM T. J. Watson Research Center, Yorktown Heights, N. Y., February 20, 1974.)

[C4] Chen, Y. C. Selective transfer analysis. IBM Res. Rep. RC 1926, IBM T. J. Watson Research Center, Yorktown Heights, N. Y., 1968.

[C5] Chow, C. K. On optimization of storage hierarchies. IBM J. Res. Develop. 18, 3 (May 1974), 194-203.

[C6] Coffman, E. G., Jr. Analysis of a drum input/output queue under scheduled operation in a paged computer system. JACM 16, 1 (January 1969), 73-90.

[C7] Control Data Corporation. 6639-A/B disk file controller reference manual. CDC Pub 60334100, 1970.

[C8] Control Data Corporation. A simulation process-oriented language (ASPOL) reference manual. CDC Pub 17314200, 1972.

[D1] Denning, P. J. Effects of scheduling on file memory operations. Proc. AFIPS 1967 SJCC, V30, 9-21.

[D2] Denning, P. J. The working set model for program behavior. CACM 11, 5 (May 1968), 323-333.

[D3] Denning, P. J. Thrashing: its causes and prevention. Proc. AFIPS 1968 FJCC, V33 Part 1, 915-922.

[D4] Denning, P. J. Virtual memory. Computing Surveys 2, 3 (September 1970), 153-189.

[D5] Denning, P. J., and Bruno, J. L. On the management of multilevel memories. Computer Science Tech. Rep. 76, Princeton University, Princeton, N. J., April 1969.

[F1] Fikes, R. E., Lauer, H. C., and Vareha, A. L., Jr. Steps toward a general-purpose timesharing system using large capacity core storage and TSS/360. Proc. 23rd. Nat. Conf. ACM, ACM Pub P-68 (1968), 7-18.

[F2] Fuchel, K., and Heller, S. Considerations in the design of a multiple computer system with extended core storage. CACM 11, 5 (May 1968), 334-340.

[F3] Fuller, S. H. Random arrival and the MTPT scheduling discipline. Proc. Fourth Symposium on Operating System Principles: Operating Systems Review 7, 4 (October 1973), 54-57.

[G1] Gecsei, J., and Lukes, J. A. A model for the evaluation of storage hierarchies. IBM Sys. J. 13, 2 (1974), 163-178.

[H1] Hogarth, J. Forthcoming Ph.D. dissertation, The University of Texas at Austin Department of Computer Sciences, The University of Texas at Austin, 1975.

[H2] Howard, John H., Jr. A large-scale dual operating system. Proc. of the ACM 1973 Annual Conference, ACM (1973), 242-248.

[H3] Howard, John H., and Wedel, Waldo M. EVENTD - UT-2D event tape summary/dump. UTEX-CC-TSN-38, Computation Center, The University of Texas at Austin, Austin, Texas, July 1974.

[K1] Keller, T. W. ASQ manual: user's guide to a program for the automatic analysis of queuing network models. TR-27, Department of Computer Sciences, The Univ. of Texas at Austin, Austin, Texas, October 1973.

[K2] Krolak, P. D. Computational results of an integer programming algorithm. Operations Research 17, 4 (July-August 1969), 743-749.

[L1] Lauer, H. C. Bulk core in a 360/67 time-sharing system. Proc. AFIPS 1967 FJCC, V31, 601-609.

[L2] Liptay, J. S. The cache. IBM Sys. J. 7, 1 (1968), 15-21.

[M1] Mattson, R. L., Gecsei, J., Slutz, D. R., and Traiger, I. L. Evaluation techniques of storage hierarchies. IBM Sys. J. 9, 2 (1970), 78-117.

[O1] Opler, A. Dynamic flow of programs and data through hierarchical storage. Proc. IFIP Congr. 1965, V1, 273-276.

[R1] Ramamoorthy, C. V. Theory of computing machines II. CS 395T class notes, The University of Texas Department of Computer Sciences, The University of Texas at Austin, Spring, 1972.

[R2] Ramamoorthy, C. V., and Chandy, K. M. Optimization of memory hierarchies in multiprogrammed systems. JACM 17, 3 (July 1970), 426-445.

[R3] Randell, B., and Kuehner, C. J. Dynamic storage allocation systems. CACM 11, 5 (May 1968), 297-305.

[S1] Scheffler, L. J. Optimal folding of a paging drum in a three level memory system. Proc. Fourth Symposium on Operating System Principles: Operating Systems Review 7, 4 (October 1973), 58-65.

[S2] Shedler, G. S. A queuing model of a multiprogrammed computer with a two-level storage system. CACM 16, 1 (January 1973), 3-10.

[T1] Teorey, T. J., and Pinkerton, T. B. A comparative analysis of disk scheduling policies. CACM 15, 3 (March 1972), 177-184.

[T2] Thornton, J. E. Design of a computer: the control data 6600. Scott, Foresman and Company, Glenview, Illinois, 1970.

[T3] Traiger, I.L., and Mattson, R. L. The evaluation and selection of technologies for computer storage systems, IBM Res. Rep. RJ 967, IBM Research Laboratory, San Jose, Cal., Feb. 8, 1972.

[V1] Varian, L. C., and Coffman, E. G. An empirical study of the behavior of programs in a paging environment. CACM 11, (1968), 471-474.

[V2] Vereha, A. L., Rutledge, R. M, and Gold, M. M. Strategies for structuring two-level memories in a paging environment. Proc. Second ACM Symposium on Operating System Principles, Princeton, N. J., (October 20-22, 1969), 54-59.

[W1] Wilkes, M. V. Slave memories and dynamic storage allocation. IEEE Trans. EC-14, 2 (April 1965), 270-271.

[W2] Wilkes, M. V. Time sharing computer systems. American Elsevier, New York, N. Y., 1968.

[Y1] Yue, P. C., and Wong, C. K. On the optimality of the probability ranking scheme in storage applications. JACM 20, 4 (October 1973), 624-633.