CHANNEL BALANCING IN A MEMORY HIERARCHY -- A CASE STUDY

Derrell V. Foster, Department of Computer Science, Duke University, Durham, North Carolina
J. C. Browne, Department of Computer Sciences, The University of Texas at Austin, Austin, Texas

I. Introduction

It appears that during every stage of development of computer systems that the demand for computer memory has increased. Efficient utilization of memory is essential to satisfy the demands for memory. The memory hierarchy concept has resulted from physical and economic considerations which make it impossible to provide unlimited storage in single memory. Several memory levels with different access times, capacities, and costs are necessary. Such a hierarchy consists of executable levels such as core memory and non-executable levels (IO devices) such as drums, disks, and tapes. It is a well-known fact that computer system performance is critically governed by the choice of a cost-effective memory hierarchy.

An important goal of the system designer is to select hierarchy management strategies which exploit the heterogeneous nature of program and data files so that files which are the least frequently accessed (have a low activity profile) are assigned to slower memories and files that are the most frequently accessed are assigned to fast memories. Such an assignment process (methodology) which satisfies not only device capacity constraints but also queuing delays associated with the devices is referred to as channel balancing. Management of executable levels has long been considered and many of the associated problems are well understood. However, management of the IO devices when viewing the computer system as a whole has been performed on a more or less heuristic basis.

After a brief discussion of a methodology for channel balancing in Section II, a case study of the assignment of the system library for the UT2D operating system is given in Section III. It should be noted that the case study uses a simulation model to provide a realistic model; however, an analytical queuing model is used to guide the simulation so that evaluation of non-optimal file assignment can be avoided.

II. Methodology

Fundamental to the methodology for channel balancing are two models: a detailed model and a gross level model. The detailed model (implemented in the case study as a simulation model) maps domain variables, notably the job characteristics, the degree of multiprogramming (potential job interference due to queuing delays), and the hardware/software characteristics of the computer system, into values of system performance metrics, say a value of system throughput as measured at the CPU.

The gross level model (implementated in the case study as an analytical queuing model) reflects the effect of file assignment through the service times of and the branching probabilities to the IO devices. For a given set of device service times, there exists a set of branching probabilities which maximizes system throughput. Thus the optimality of a file assignment can be estimated by how accurately the model yields the values of this set. A knowledge of the characteristics of rotating devices reveals that mean service times are effected by file assignment somewhat less than are the branching probabilities. Note also that the identity of the files is not maintained in the gross level model.

A methodology for channel balancing is given by the following iterative procedure:

(i) Select some initial file assignment.
(ii) Evaluate the detailed model for the service times of the IO devices.
(iii) Evaluate the gross level model for the branching probabilities which maximize throughput using the service times determined in step (ii).
(iv) Select a new file assignment whose accumulative frequency of file request on an assigned device satisfy the optimal branching probability constraints determined in step (iii) while not violating device capacity constraints.
(v) Iterate on steps (ii) - (iv) until no changes in file assignment occur.

A more detailed description of this methodology is given in [1].

III. The UT2D Peripheral Processor Library -- A Case Study

3.1 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 smaller 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 [2,3].

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 system. 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.

3.2 The PPU Subsystem

The PPU subsystem interconnection topology is given in Figure 1. The system consists of four servers, central memory, ECS, a disk unit, and a PPU. Both the simulation model and analytical queuing model corresponding to the topology are given in Figure 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).

3.3 Simulation Model Parameterization

The following parameters are assumed to accurately and sufficiently characterize the behaviour 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.

3.3.1 Activity Profile. If the jobs themselves are requests for the loading and executing program files, then the job characteristics are commonly given in an activity profile, one entry per file. The activity profile is composed of four 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 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 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 [4]. The request frequency parameter is simply a count of the number of times that a given PPU transient program is located. The instructions executed/request parameter is obtained from the mean time between consecutive PPU transient program locations.

3.3.2 Degree of Multiprogramming. Another major parameter of the simulation 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.

3.3.3 Hardware Characteristics The hardware characteristics assumed by the simulation 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 bits) 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

3.4 Simulation 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.

3.5 Results Obtained Through Channel Balancing

3.5.1 Throughput as a Function of Memory Capacity. The table in Figure 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. 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 capacitieshfor 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 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.

3.5.2 Program Assignment. Assignment of the programs in the activity profile of Figure 3 to be IO devices is given in Figures 5a and 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.

IV. Summary

Values for throughput and a corresponding program assignment as a function of memory capacity of the IO devices are generated utilizing a channel balancing methodology. This automated technique allows the system designer to concentrate on major performance characteristics (such as throughput) without unduly being burdened by system details (such as program assignment). The methodology extends previous work by including queuing delays for the IO devices in a memory hierarchy, and optimization of total system throughput. An implementation of this methodology uses a simulation model to provide a realistic model; however, an analytical queuing model is used to guide the simulation so that rapid evaluation of program assignment is possible.

V. Acknowledgements

This research was sponsored by the National Science Foundation under Grant GJ-1084 and by the Department of Computer Sciences of the University of Texas at Austin.

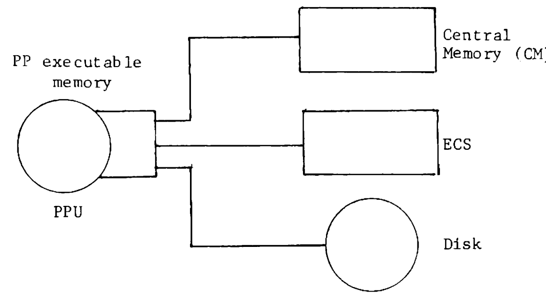
References

[1] Foster, Derrell V., "File assignment in memory hierarchies", (Ph.D. Dissertation), The University of Texas at Austin, August, 1974.

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

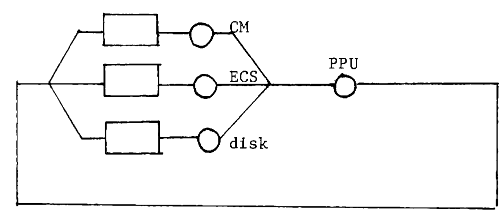
[3] Thornton, J. E., Design of a computer the control data 6600, Scott, Foresman and Company, Glenview, Illinois, 1970.

[4] 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.



PPU Subsystem
Interconnection Topology

Figure 1



PPU
Simulation and Analytical Queuing Model

Figure 2

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

Activity Profile
of PPU Transient Programs

Figure 3

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

N	J	ID	F	T	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ REQ	COMPLTED LOAD REQ	CURRENT LOCTN
1	CIO	P	22960.0	3	124	124	.2910382	.2948231	1475	1475	FMEM
2	2WD	P	11861.0	93	85	85	.1503486	.1511093	756	755	ECS
3	2RD	P	11289.0	95	82	82	.1430980	.1341195	671	671	ECS
4	CPU	P	3983.0	9	270	270	.0504880	.0521687	261	261	ECS
5	1DB	P	3853.0	67	153	153	.0488402	.0517689	259	259	ECS
6	2MT	P	3432.0	119	458	458	.0435036	.0449730	225	225	ECS
7	2PD	P	3367.0	46	179	179	.0426797	.0409754	205	205	ECS
8	1PS	P	2208.0	3	138	138	.0279883	.0285929	143	143	ECS
9	2WM	P	2102.0	16	191	191	.0266447	.0275834	138	138	ECS
10	RFL	P	1396.0	69	120	120	.0176955	.0195882	98	98	ECS
11	OPE	P	1338.0	7	72	72	.0169603	.0183890	92	92	ECS
12	2DF	P	875.0	4	31	31	.0110914	.0105936	53	53	ECS
13	SNP	P	760.0	6	55	55	.0096337	.0123926	62	62	ECS
14	MSG	P	745.0	3	38	38	.0094435	.0075954	38	38	ECS
			TOTALS:	540	1096	1996	.8894537	.8946632	4476	4475	
			ACCUMLATVE TOTALS:	540	1996	1996	.8894537	.8946632	4476	4475	

TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 248

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

PPU Transient Program Assignment to ECS

Figure 5a

N	J	ID	F	T	R	V	ACTUAL REL FREQ	OBSERVED REL FREQ	OBSERVED OBJ REQ	COMPLTED LOAD REQ	CURRENT LOCTN
15	PFM	P	1214.0	28	304	304	.0153885	.0155906	78	78	DISK
16	EPR	P	1196.0	16	373	373	.0151603	.0167899	84	84	DISK
17	2TS	P	910.0	100	350	350	.0115350	.0131921	66	66	DISK
18	1AJ	P	900.0	4	166	166	.0114083	.0109934	55	55	DISK
19	1RJ	P	586.0	941	230	230	.0074281	.0055966	28	28	DISK
20	1PL	P	585.0	63	148	148	.0074154	.0063962	32	32	DISK
21	LDR	P	578.0	259	199	199	.0073267	.0057965	29	29	DISK
22	1SJ	P	567.0	280	181	181	.0071872	.0059964	30	30	DISK
23	1SS	P	392.0	174	282	282	.0049689	.0035978	18	18	DISK
24	2PU	P	352.0	49	266	266	.0044619	.0035978	18	18	DISK
25	PCC	P	174.0	14	45	45	.0022056	.0027983	14	14	DISK
26	3AJ	P	165.0	54	101	101	.0020915	.0013992	7	7	DISK
27	1TD	P	148.0	742	116	116	.0018760	.0019988	10	10	DISK
28	2SP	P	140.0	60	336	336	.0017746	.0029982	15	15	DISK
29	1CJ	P	124.0	56	59	59	.0015718	.0013992	7	7	DISK
30	2FE	P	117.0	32	10	10	.0014831	.0005996	3	3	DISK
31	2TJ	P	116.0	42	26	26	.0014704	.0011993	6	6	DISK
32	2JE	P	109.0	21	15	15	.0013817	.0013992	7	7	DISK
33	2AM	P	97.0	36	62	62	.0012296	.0003998	2	2	DISK
34	3EA	P	60.0	171	180	180	.0007606	.0003998	2	2	DISK
35	RCC	P	48.0	16	39	39	.0006084	.0009994	5	5	DISK
36	2E2	P	46.0	50	96	96	.0005831	.0005996	3	3	DISK
37	2E1	P	41.0	25651	95	95	.0005197	.0003998	2	2	DISK
38	CLO	P	28.0	17	12	12	.0003549	.0005996	3	3	DISK
39	1SR	P	28.0	32	132	132	.0003549	.0005996	3	3	DISK
				----	----	--------	----------	----------	---	---	
			TOTALS:	28908	3823	3823	.1105463	.1053368	527	527	
	ACCUMLATVE TOTALS:			29448	5819	5819	1.0000000	1.0000000	5003	5002	

TOTAL RECORD SIZE (R) OF CURRENTLY LOADED OBJECTS: 0

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

PPU Transient Program Assignment to Disk

Figure 5b