SYSTEM PERFORMANCE MEASUREMENT AND OPTIMIZATION

JIM SQUIRES, H-P Systems Engineer, Fullerton, Ca.
ED SPLINTER, H-P Systems Engineer, Los Angeles, Ca.

Obtaining optimum performance from a computer system is often critical to the success of an installation. This is particularly true today when data processing managers are being ask to produce more with little or no increase in their budget.

When most users comment about a system's performance they are really stating how well the system meets their expectations. This means that what is felt to be a performance problem might well turn out to be an expectation problem. If users fail to consider the strengths and limitations of the system while designing application programs great disappointment can result.

Fortunately performance measurement tools formerly used only at the factory have matured and are now being distributed to the field for SE use. With these tools now available at a point closer to the customer, performance problems are being addressed more quickly and in many cases with impressive results.

On the following pages is a representative report based on one of the machines where performance was judged by the users to be unsatisfactory. The report is presented to the customer during a meeting that usually lasts in the neighborhood of two hours. At that time attention is focused on the areas where improvements in performance can be realized as quickly as possible. At all times it must be remembered that the object is to optimize the combination of the computer system and its users not just the system.

INTRODUCTION	SECTION I

INTRODUCTION

The purpose of this report is to present an analysis of the performance of your HP3000. This information should help in answering questions such as:

1. Is system response time being restricted by CPU, memory or disc contention?
2. Are any programs unexpectedly dominating the mix?
3. Can additional applications and/or users be added to the system without adversely affecting response?
4. What might be done to improve system performance?

The data presented here was obtained from a trace of system activity collected with the event monitoring facility. A record of each occurence of selected events is written to tape for subsequent analysis. For this report the primary events monitored are associated with memory management activity, process dispatching and IO device activity.

Since data is collected with a monitor using software traps, the monitoring activity necessarily has an effect on the performance of the system. Experience indicates, however, that the results are skewed only slightly and in most cases is undetectable. In any case the information obtained gives one a far greater insight into the system's activity than is obtainable in any other way currently available.

Raw data is usually collected for a period of time much longer than that chosen for detailed analysis. On a heavily loaded system about 500,000 events are recorded on the tape each hour activity is monitored. Since detailed analysis is quite time consuming the tape is scanned for a general picture of the activity recorded. A 15-30 minute 'window' is then chosen for detailed analysis.

The operating system of any computer is designed to manage the system's resources, principally, the processor, main memory and disc resolving conflicts arising from competition amoung the community of users. When demand for any resource approaches the capacity, the management task becomes difficult causing system efficiency to decline.

Performance of a system has to be discussed in the context of the system workload. The programs which make up this workload can be characterized by the type and amount of system resources required for their execution.

In the following sections this report moves from the general to the specific in its investigation of the utilization of the three principal resources mentioned above. First, utilization from an overall point of view is discussed. Then a summary of information by program for all jobs and sessions is presented. Next is a detailed report for each program that was found to be a significant resource user. The section on conclusions and recommendations provides a summary of the significant bottlenecks in the system and suggests ways to improve system performance.

DATA COLLECTION

Period Monitored: Mon, Jun 26, 1978 1:32 - 2:23pm

Total number of events recorded: 461,301

Window chosen for analysis: 1:53 - 2:13pm (1200 secs)

Unless otherwise noted
ALL TIMES IN SECONDS
ALL LENGTHS IN BYTES

OVERALL CENTRAL PROCESSOR UTILIZATION

A fairly good first approximation of how well a system is performing is given by noting the amount of time the CPU spends in each of four states:

1. CPU busy - the time during which some process was executing;
2. Waiting for swaps - the time during which the memory manager (MAM) is waiting for a disc I/O to complete and no other process has sufficient memory resources to run;
3. Waiting for disc I/O - the time during which a process other than MAM is waiting for a disc I/O to complete and no other process is requesting the CPU;
4. Idle - the time during which no process is requesting the CPU and no process is waiting for I/O to terminate.

CPU STATE	Percent of Window	Total Mins
Busy	74.57	14.9
Waiting for swaps	15.31	3.1
Waiting for disc	7.87	1.6
Idle wait	2.25	0.5

FIGURE 1-1. CPU usage during the window. The average CPU busy interval was 14 ms and the average idle time was 5 ms. These two figures were distorted by the program IDLE.

The CPU busy time can be broken down as follows:

Memory manager	10.36%
Other MPE processes	2.00
The program IDLE	≈30.52
Other user processes	31.69

Note that the CPU was being used by the memory manager or was being held for swapping 25.67% of the time. This indicates that the memory manager is having considerable difficulty meeting the requests for main memory.

OVERALL MEMORY UTILIZATION

As your system is presently configured the resident portion of MPE uses 84,464 bytes of memory leaving 439,824 bytes of 'linked' memory for swapping. The size of resident memory is, to some extent, controlled by responses to configuration questions while doing a SYSDUMP.

In analyzing the utilization of memory it is useful to note whether the allocation was for code or data and if for code whether it came from a program file or a segmented library (SL).

Segment Type	Percent Memory	Average Alloc	Average Presence	Num of Swaps	Overlays per Swap
DATA	30.75	3679	8.13	4912	0.741
SL	44.23	5272	12.47	3331	1.412
PROGRAM	8.46	4393	18.02	550	0.391
Totals	83.44	4327	10.41	8793	0.973

FIGURE 2-1. Memory allocation information. This data applies only to linked memory and does not include any segments allocated prior to the start of monitoring.

The principal concern here is how much swapping went on and how many segments currently in memory had to be overlayed to make room for the new one. The number of swaps shown here indicates a higher than desirable rate of swapping. There were over 7 swaps per second. It is possible to average about 30 disc I/Os per second on the HP3000. This means that about 25% of the maximum possible disc activity was used for swapping.

OVERALL DISC ACTIVITY

For best performance, disc I/O requests should be evenly distributed over the available drives to reduce arm contention and seek times.

Drive	Request Count	Percent of Total	Transfer Length	% Busy	Seconds Between Requests
1 R	11171	40.1	2924	25.65	0.107
1 W	6613	23.7	3100	15.52	0.181
2 R	1228	4.4	2816	2.80	0.974
2 W	871	3.1	1887	1.54	1.378
3 R	1596	5.7	2261	3.57	0.752
3 W	808	2.9	1169	1.55	1.485
4 R	1146	4.1	2244	2.19	1.045
4 W	901	3.2	2003	1.65	1.330
12 R	1094	3.9	3044	2.62	1.093
12 W	607	2.2	1673	0.99	1.973
13 R	1164	4.2	1632	2.15	1.027
13 W	661	2.4	1388	0.93	1.809
	27860				

FIGURE 2-2. Global disc activity. The top line for each device applies to reads, bottom line to writes. During the window 74.35 million bytes of data were transferred between disc and memory. Swapping traffic accounted for 57.04 million bytes or 76.7% of the total. Each second an average of 23.22 disc I/O requests were made.

OPERATING SYSTEM DISC REQUESTS

Certain user activities cause the system to access disc storage. It is useful to identify these activities and tabulate their respective I/O loads. The memory manager is almost always the system process which uses disc most heavily. All MAM requests are associated with swapping. The LOADER accesses program files and SL files to resolve external references when the :RUN command is issued. Other system processes which access disc memory include DEVREC (to verify signon information) and LOG (for system logging, if enabled).

DRIVE	SYSTEM PROCESSES MAM	LOADER	OTHER	USER PROCS	TOTAL
1 R	7862	853	72	2384	11171
1 W	5466	43	2	1102	6613
2 R	65	1	123	1039	1228
2 W		2	43	826	871
3 R	480	95	12	1009	1596
3 W		2	5	801	808
4 R	168	74	31	873	1146
4 W		65	6	830	901
12 R	111	26		957	1094
12 W		24		583	607
13 R	108		15	1041	1164
13 W			29	632	661
	14260	1185	338	12077	27860
	51.18%	4.75%	1.21%	43.35%	

FIGURE 2-3. Operating system disc access requests. The top line for each device applies to reads and the bottom line is for writes.

During the monitoring window of a system it is possible for each user to run one or more programs. Figure 3-2 shows which programs were run by each user. Also shown are those programs used by the operating system. Included for each user's program is the CPU, memory and disc resources used. Figure 3-1 contains information to help identify users.

The system programs listed are run on behalf of users without their knowledge or intervention. Normally, all of these programs make small demands on system resources. Since most of these programs are run at a much higher priority than user programs, their impact is quickly felt when they become heavily used. The fact that MAM used over 10% of the CPU is an immediate indication that the operating system is have trouble meeting the demand for memory. There is not enough real memory to efficiently handle all the requests.

Note the impact that the COBOL compile (#J5) had on the system. The combined compile and prep used 21% of the CPU and 28% of the memory during the 6 minutes that the operation took. When response times are a problem, COBOL compiles should be kept to an absolute minimum.

The data clearly shows the impact that the A4000 processes have. With only a couple of exceptions, those processes with more than 190 swaps are associated with your application program. The exceptions are significant since they include COBOL and the EDITOR. The EDITOR process during Session 41 ran 18 minutes using over 7% of the available memory and over 4% of the CPU during this time.

JOBNUM	JIN	INTRODUCED	JOB NAME
#S29	20	MON 1:33P	MANAGER.SYS
#S34	52	MON 1:35P	A40,RON.FMS
#S35	40	MON 1:35P	OSSUSER.FMS
#S36	23	MON 1:37P	A40,OSSUSER.FMS
#S37	27	MON 1:37P	A40,OSSUSER.FMS
#S38	26	MON 1:37P	A40,OSSUSET.FMS
#S39	28	MON 1:38P	A40,OSSUSER.FMS
#S40	25	MON 1:40P	A40,OSSUSER.FMS
#S41	53	MON 1:42P	GEORGE.FMS
#S48	32	MON 1:47P	A40,OSSUSER.FMS
#S50	31	MON 1:49P	A40,OSSUSER.FMS
#S52	30	MON 1:50P	A40,OSSUSER.FMS
#S53	34	MON 1:50P	A40,OSSUSER.FMS
#S55	22	MON 1:52P	A40,OSSUSER.FMS
#S56	51	MON 1:56P	ANNIE.FMS
#S57	23	MON 1:57P	A40,OSSUSER.FMS
#S58	21	MON 1:57P	OP.FMS
#S60	27	MON 1:59P	A40,OSSUSER.FMS
#S61	46	MON 1:59P	A40,OSSUSER.FMS
#S64	35	MON 2:02P	A40,OSSUSER.FMS
#S65	40	MON 2:03P	OSSUSER.FMS
#S67	50	MON 2:04P	RONK.FMS
#S68	41	MON 2:04P	A40,OSSUSER.FMS
#S71	55	MON 2:12P	TOM.FMS
#J 1	10	MON 1:34P	IDLE,DAN.FMS
#J 5	10	MON 1:50P	WCOMPILE,ANNIE.FMS
#J 6	10	MON 1:56P	MANAGER.SYS

FIGURE 3-1. Session and job identification.

Figure 3-2 on the next two pages contains summary information about each program that was running anytime during the 1200 second window. The number of seconds that the program was observed is shown in column two. CPU usage is shown as a percentage of the seconds observed. Memory used is a percentage of memory available during the time observed. Column five indicates the average size of all segments (code and data) allocated for the program. The disc IO count in column six applies only to IOs associated with files opened by the program. The swap count includes all memory manager IOs caused by this program including the initial allocation of each segment. Overlays indicate how many segments already in memory had to give up memory when the average segment for this program was made present.

J/S# PROG	SECS SEEN	%CPU USED	%MEM	AVG SEG	DISC I/Os	SWAPS	OVER LAYS
SESSIONS							
29 COMMANDS	1197	.06	.37	3607	11	81	.19
34 SEGDVR	182	.33	1.21	2257	2	105	.13
SEGPROC	185	3.78	2.78	3698	657	78	.69
COMMANDS	1117	.47	.91	3080	300	281	.30
35 A4000	585	3.02	12.67	4769	260	535	1.39
COMMANDS	617	.05	.15	3128	16	24	.00
36 A4004	241	1.26	3.00	4570	98	114	.32
COMMANDS	259	.11	.55	3015	16	32	.00
37 COMMANDS	338	.26	.41	2957	18	56	.34
A4000	338	3.16	12.09	4830	224	456	1.40
38 A4004	1195	1.39	3.45	6177	528	278	1.66
39 A4002	1197	2.62	9.68	5304	1124	559	1.35
40 A4003	1200	.53	1.94	5154	113	179	1.55
41 EDITOR	1119	4.14	7.35	3722	1601	538	.40
COMMANDS	1143	.04	.06	3241	14	39	.33
EDITOR	54	2.59	9.38	4214	68	49	.63
48 A4000	1188	.66	4.67	5898	134	352	1.63
50 A4001	1196	.98	4.49	5403	405	308	1.24
52 A4000	1165	.93	4.37	5402	299	346	1.62
53 COMMANDS	1200	.11	.34	3227	70	79	.22
A4004	1197	.01	.03	2841	0	13	.15
54 COMMANDS	205	.30	.69	2691	24	60	.25
55 COMMANDS	4	1.86	4.29	1130	3	4	.00
A4003	626	.90	3.38	4404	247	196	1.14
56 FCOPY	106	.56	2.96	3688	7	48	.90
FCOPY	301	.65	4.56	3890	15	122	.52
COMMANDS	852	.35	.54	2673	98	127	.40
57 COMMANDS	29	1.22	4.11	2695	24	25	.24
A4004	892	.32	.41	3437	138	61	1.02
58 COPYOP	195	1.33	3.67	3602	93	72	.47
COMMANDS	262	.52	1.17	3150	42	59	.85
59 COMMANDS	15	3.01	1.93	2445	28	16	.56
60 A4000	737	2.96	5.81	5219	1165	357	1.96
COMMANDS	108	.35	1.00	2634	24	27	.04
61 A4004	30	3.15	11.09	3207	12	43	.46
COMMANDS	75	.78	2.08	2542	36	32	.53
A4004	64	4.29	8.00	4270	139	58	1.00
62 COMMANDS	16	2.76	4.97	2595	28	26	.65
63 COMMANDS	13	3.29	5.23	2402	28	30	.27
64 A4003	611	1.27	4.76	4896	298	211	1.21
COMMANDS	26	1.50	3.47	2727	24	16	.19
65 COMMANDS	37	1.16	5.31	2485	22	27	.22
A4000	522	2.09	5.51	4041	272	323	.80
66 COMMANDS	26	2.02	1.68	2634	28	31	1.00

FIGURE 3-2A. Summary information of each program seen during the window. COMMANDS refers to the command interpreter.

J/S# PROG	SECS SEEN	%CPU USED	%MEM	AVG SEG	DISC I/Os	SWAPS	OVER LAYS
67 EDITOR	219	10.13	4.70	4067	381	81	.32
COMMANDS	302	.37	.81	3146	46	65	.31
EDITOR	215	10.63	6.51	3948	437	101	.26
68 COMMANDS	28	1.14	2.93	2789	24	16	.13
A4003	316	2.77	8.87	4925	278	196	1.31
69 QUERY	275	2.13	5.34	5164	179	148	.97
COMMANDS	399	.33	1.42	3013	70	110	.18
70 COMMANDS	136	.54	2.03	2980	31	56	.50
FCOPY	69	6.16	6.59	3438	65	43	.93
71 COMMANDS	20	1.67	4.19	2813	26	25	.68
EDITOR	41	4.09	4.90	4158	67	48	.83
JOBS							
1 IDLE	1143	30.52	1.38	2421	9	26	.00
5 COBOL	299	16.47	19.93	8650	493	224	2.98
COMMANDS	358	.26	.44	3060	48	40	1.17
SEGPROC	69	4.44	7.60	3436	202	57	.91
6 COMMANDS	33	4.48	6.07	3297	58	42	.33
8 FORTRAN	86	10.81	9.16	3838	440	89	.58
COMMANDS	179	.60	3.12	3053	43	81	.44
FORTRAN	197	4.30	9.79	4513	426	68	1.28
SYSTEM							
PROGEN	287	.27	.47	4210	16	20	1.05
MAM	1200	10.36	.00	0	14260	0	.00
IOSYS	1199	.06	.20	2014	0	76	.01
IOMSG	1165	.04	.27	1880	0	133	.02
LOG	1198	.02	.25	2415	63	91	.07
UCOP	1200	.15	.52	1945	4	174	.10
DEVREC	1198	.11	.47	2424	68	121	.05
PRIMSG	1197	.02	.05	1196	0	29	.00
LOAD	1132	1.52	.29	4870	1185	45	.73
SPOOLER	443	.17	.27	3117	14	14	1.36
SPOOLER	1061	.72	1.13	4424	173	101	.52

FIGURE 3-2B. Summary information of each program seen during the window. COMMANDS refers to the command interpreter for the job or session.

Programs normally spend very little time actually using the CPU. Most of the time is spent waiting for some event to terminate. Three events usually account for the majority of the wait time.

(1) User requested I/O,
(2) Absent code or data segment,
(3) Human think time at a terminal.

Of course the third item usually doesn't apply to batch programs. In this case waiting for a higher priority process to give up the CPU is the third significant event.

Interactive programs may also be held up waiting for terminal output. This is caused by writing large amounts of information (i.e. large forms) to the screen in block mode. Adding more terminal buffers to the system configuration will sometimes help. Occasionally programs are seen with significant wait due to database or file locking.

Once the events dominating the wait time have been identified it may be possible to improve performance of individual programs and thus the system as a whole. When the CPU is the limiting resource, the solution is usually an additional computer or another more powerful.

When absent segments are responsible for most of the wait, performance can be improved by adding more real memory to the system. This condition can be caused by segments which are excessively large (over 10,000 bytes). In this case reducing segment sizes may improve performance to a satisfactory level. The cost of modifying programs to accomplish this must be balanced against the cost of the required additional memory. Frequently adding memory is the most cost-effective solution.

User disc I/O wait time can normally be reduced only through reduction of I/O requests by the program. In a few instances moving files between drives to balance arm contention may help.

Locking waits can often be reduced by carefully rethinking where and when lock requests are issued. Applications locking multiple files (or databases) can make reduction of locking contention very difficult.

PROCESS/ SESSION	SECONDS OBSERVED	USING CPU	ABSENT SEGMENT	USER DISC IO	FILE LOCK	TERM READ
		%	%	%	%	%
A4000/35	585	3.0	21.1	1.6	2.3	66.8
37	338	3.2	30.3	2.4	14.8	45.7
48	1188	0.7	6.4	0.6	17.4	74.4
52	1165	0.9	6.2	1.0	17.4	70.7
60	737	2.9	13.9	5.7	10.5	53.0
65	522	2.0	19.2	1.5	15.2	58.6
A4001/50	1196	1.0	3.2	1.6	13.4	66.3
A4002/39	1197	2.7	9.6	3.5	22.3	49.8
A4003/40	1200	0.5	3.5	0.4	0.7	78.2
55	626	0.9	6.5	1.6	13.8	74.7
64	611	1.3	8.1	1.6	9.6	78.5
68	316	2.7	13.9	2.9	15.3	52.4
A4004/36	241	1.3	9.8	1.5	0.0	83.5
38	1195	1.4	3.3	2.3	10.7	72.5
53	1197	0.0	0.1	0.0	0.0	99.0
57	892	0.3	1.0	0.4	0.0	97.8
61	30	3.2	27.9	1.4	0.0	27.2
61	64	4.4	33.2	5.9	0.0	42.8

FIGURE 4-1. Program wall-time distribution. This chart shows what was happening to programs during the time each was being observed. For instance, during the 585 seconds that program A4000 (#S35) was visible within the window it spent 3% of that time executing, 21.1% of the time waiting for a segment to be made present before execution could continue, 1.6% of the time waiting for file I/Os to complete, 2.3% of the time for a data-base lock and 66.8% of the time waiting for response from a terminal read.

The following processes spent a significant amount of time waiting for 'blocked I/O'. This is caused by writing to a terminal when termbufs are unavailable or by nobuf I/O. The largest factor in this case is probably due to the nobuf I/O calls issued by IMAGE.

A4000/60	13.64%
A4001/50	11.70
A4002/39	16.57
A4003/40	14.41
A4003/68	10.06

Program: A4000 #S35 OSSUSER.FMS

Observed: 13:36:40 - 14:02:45 585.4 secs

WAITING FOR:	OCCURRENCES PRCNT	OCCURRENCES COUNT	SECS BETWEEN OCCURENCES	AVERAGE WAIT	CPU BETWEEN
ABSENT SEG	45.15	521	1.123	.237	0.034
DISC I/O	22.70	262	2.233	.035	0.067
TERM READ	16.90	195	2.996	2.005	0.087
HIGHER PRI	12.31	142	4.120	.021	0.124
MPE RESOURCE	1.47	17	34.240	.390	0.994
TERM WRITE	1.21	14	41.522	1.480	1.204
DATA BASE LOCK	.26	3	180.617	4.562	5.281

MEMORY AND SWAPPING LOAD:

	MEMORY USED PRCNT	MEMORY USED AVG SIZE	TIME IN MEMORY AVERAGE	TIME IN MEMORY TOTAL	SWAPS	OVER LAYS
STACK97	1.008	7816	15.099	332.1	22	1.772
DSEG100	.002	7872	.701	.7	1	2.000
DSEG101	.002	7640	.668	.6	1	5.000
DSEG104	.006	7552	2.149	2.1	1	3.000
DSEG94	.004	5440	1.935	1.9	1	2.000
DSEG74	.120	5026	6.621	59.5	9	.333
DSEG99	.185	2216	7.958	214.8	27	.000
DSEG84	.177	1464	10.385	311.5	30	.000
Avgs for 28 data segs	2.015	2353	6.655		238	.256
Avgs for 44 SL segs	10.643	6714	13.472		296	2.311
Avgs for 1 prog seg	.011	3904	7.357		1	.000

FIGURE 4-2. Process detail. This illustrates the level of detailed information available for each process active within the window. Events which caused the process to wait are listed at the top in order of number of occurrences. Note that this process used only 67 ms of CPU time between each disc I/O which occurred on the average every 2.2 seconds.

While the data here indicates a terminal response time of 991 ms (time between reads minus wait for read) actual response time was around 15 secs. The difference is caused by multiple reads being issued for a formatted screen.

CONCLUSIONS AND RECOMMENDATIONS	SECTION V

The data presented in this report indicates a couple of reasons for your reduced response time. The memory manager is having trouble meeting demand for main memory. This is shown by the fact that MAM used over 10% of the CPU during the window. A swap rate of over 7 per second is another indication.

The memory contention problem can be reduced to some extent by reducing the size of the five large SL segments used by you application program. You should be able to identify these using the individual program data sheets in Section IV.

The second problem is database locking contention. Data in Section IV shows that most executions of your application spent about 15% of the time waiting to lock databases. This problem was particularly severe for A4002 (#S39) which spent over 22% of the time waiting for database access. No suggestions can be made here since a solution, if available, would require intimate knowledge of the application.

The information presented in Section III shows the impact of program development activity. Note the CPU and memory usage during executions of EDITOR, FCOPY and COBOL. When response times become unbearable, it may be necessary to curtail on-line program development.

During the 50 minutes that data was being collected 54 log-ons occurred. Since the log-on process places a heavy load on the system, even though for a short time, an attempt should be made to reduce this activity. At least 29 sessions lasted less than one minute.

An effort should be made to keep at least 20,000 free sectors on the system disc. At least 12-15% of the total disc space should be free at all times. Theory and experience both show that when the average utilization of any resource approaches the capacity, a large performance degradation results.

Data in Section IV seems to indicate that when the memory and locking bottlenecks are removed, it is unlikely that another "bottleneck" will be uncovered. Almost certainly the CPU will not restrict you for the foreseeable future. Disc activity level is low enough to suggest no problem will occur here either.

There are several programs more or less available that in some way provide information pertaining to performance. Many are contributed and thus may or may not execute properly with the lastest release of MPE. The programs marked as contributed are in general circulation but have not been formally placed in the contributed library.

1. MONITOR — the combination of a software monitor built into MPE and a data reduction program. This is the most complete performance tool available. Output from the reduction program requires careful interpretation. Dedicated tape controller and drive required. Available only to SEs.

2. SAMPLER* — used to identify those sections of code which are executed most frequently. Requires installation of an additional clock board on the system to be sampled. Dedicated tape controller and drive required. Available only to SEs.

3. TRACER* — measures segment boundary crossings to determine which segments are referenced and how often. Can not be used with COBOL programs. Dedicated tape controller and drive required. Available only to SEs.

4. TUNER2 — shows current and maximum use of several system tables. Used to determine whether configuration parameters have been correctly chosen. Contributed.

5. OVERLORD — useful in determining who is executing what program and the corresponding stack size. SOO (Son of Overlord) provides the same functions. In the contributed library.

6. SHOWVM — indicates, at a gross level, how real and virtual memory is being used. Helpful in determining how much memory a particular program uses. In contributed library.

7. SHOWQ — a system command which displays information about process scheduling subqueues. If the rightmost of the three columns displayed grows longer than the center column, then the system has insufficient real memory for the current load.

* General availability currently scheduled for early 1979.

8. FREE2 — displays the amount of free space on all system disc packs and indicates how fractured the space is. Badly fractured disc space can cause a considerable performance degradation. HP supported system utility.

9. SYSINFO — prints system configuration information without doing a SYSDUMP. Contributed.

10. PROGINFO — prints stack size, segment size and external reference information about program files. An extended version of PROGSTAT. Contributed.

11. KSAMUTIL — displays blocking factors and intrinsic call counts associated with KSAM files. HP supported utility delivered with KSAM.

12. LOG FILES — contain information associated with log-ons, file closes and I/O transfer counts by process. Some contributed programs available to reduce information in these MPE generated files.

13. LISTFXX — used to carefully track use of disc space. Indicates location of first extent of each file and date of last access in addition to other information. When used in conjunction with FREE2 the amount of recoverable lost disc space can be calculated. In the contributed library.

14. DBDRIVER — used to quickly obtain information about an IMAGE database such as the size of the DBCB and individual transaction times for a single user. Distributed with IMAGE but is unsupported.

15. IDEA — helps determine performance characteristics of an IMAGE based application. Can measure the effect of multiple users. Timings apply only to the IMAGE response times, not application processing times. Contributed.

16. DBSTAT — determines length of synonym chains in IMAGE master data-sets. Long synonym chains can cause dreadful performance degradation. Contributed.

17. LISTDIR2 — used to determine location of the first extent of disc files. HP supported system utility.

18. LISTF — The '-1' option displays the file label which contains the address of each extent of the file. Output can be written to a disc file for subsequent programatic processing. HP supported MPE command.

MPE USE OF MAIN MEMORY	APPENDIX B

The amount of memory reserved exclusively for MPE is referred to as fixed memory as opposed to linked memory for which all processes compete. The size of fixed memory can have a substantial effect on performance of machines with less than 512kb of memory.

FIXED MEMORY GENERAL LAYOUT	SIZE BYTES		ITEM
Driver linkage table	256		
CST block table	132	*	30
Bank table	192		
Job process count table	88		
System global area	768		
Data segment table	3072	*	4
Code segment table	1536	*	2
Code segment table extension	4000	*	3
Process control block table	4096	*	5
Interrupt control stack	1088	*	10
Terminal buffers	4112	*	7
I/O queue	2832	*	6
Interrupt linkage/Device info table	6520		
System buffers	4664	*	8
Working set table	4464	*	30
Memory management table	3840	*	9
Virtual bit map	512		
Virtual disc space locator	768		
Logical-physical device table	528		
Timer request list	456	*	12
Job cutoff table	424		
System internal resource table	440		
Breakpoint table	520	*	13
Memory management code	10200		
Miscellaneous system routines	3272		
System clock & timer req list code	3304		
Resident IO code	12064		
Internal interrupt handler code	1880		
Dispatcher code	1376		
Disc driver code	2424		
Tape label input buffer	552		
Miscellaneous areas	5000		

FIGURE B-1. Items in fixed memory of the 2Mb system at the Fullerton HP Technical Center listed in approximately the sequence they actually appear in memory. Lines with asterisks indicate segments whose size is directly affected by responses given at SYSDUMP time. The item numbers are references to Appendix C of the System Manager Manual.

SEMI-RESIDENT SYSTEM CODE

In addition to the 60-90kb of fixed memory several MPE segments are referenced so frequently that they tend to spend much of the time in memory. On the average 3000, MPE is probably holding at least 150kb of memory at all times. Since this is not affected by the total amount of memory on the system adding more memory to systems in the 256-512kb size can substantially improve performance.

The following system code segments tend to spend at least 75% of the time in memory:

FILESYS1	8952	FREAD, FWRITE
FILESYS1A	5400	FILESYS support routines
FILESYS2	5624	FPOINT, FCONTROL, FUPDATE
FILESYS5	4208	FILESYS support routines
FILESYS6	3472	FILESYS support routines

Other code segments which spend over 60 percent of the time present:

ALLOCUTIL	5848	Device allocation utilites
PINT	3048	CALENDAR, CLOCK, Process support
DATASEG	5040	Data segment handling routines
CHECKER	1536	GETPRIVMODE, Intrinsic errors
UTILITY1	3208	ASCII, BINARY, WHO, READ, PRINT
IOTERM0	5128	Terminal driver
	23808	Total bytes

Some MPE data segments which tend to stay in memory are listed below with a typical size in words.

UCOP request queue	208	
LDEV table	2928	
Disc directory seg	2056	
Job master table	1024	
Volume table	200	
FMAYT	528	
Process/job xref	264	
	7208	Total bytes

Program	Total Words	Num Segs	Global DB-QI	STACK QI-ZI	MAX DATA	SL Segs	SL Words	Tot Segs
APL	138520	49	6883	1024	31000	21	48012	70
BASIC	41564	24	496	800	31000	24	53344	48
BASICOMP	33184	18	1146	800	30000	15	37756	33
COBOL	84892	35	3953	2000	32000	18	41212	53
EDITOR	33380	15	995	4600	8000	17	39680	32
FCOPY	16080	5	4893	800	31000	19	50304	24
FORMAINT	7088	2	1722	800	20000	10	23199	12
FORTRAN	45188	21	1701	2500	32767	16	36124	37
FREE2	696	1	4172	800	DFALT	10	23048	11
LISTDIR2	8936	4	795	800	8192	16	42460	20
LISTEQ2	1012	1	2487	800	15000	5	12800	6
MERGE	2856	1	2	800	15000	15		16
MPMON	2956	1	5524	800	DFALT	15		16
MRJE	20488	7	3221	800	DFALT	19	42540	26
MRJEMON	1860	2	394	1100	5200	23	52208	25
MRJEOUT	1816	2	2306	1024	DFALT	12	29352	14
QUERY	42820	20	4174	1300	11000	28	62684	48
RESTORE	1948	1	2413	800	DFALT	8	20404	9
RJE	5400	8	927	1000	DFALT	24		32
RPG	55396	22	3146	800	32000	15	37604	37
SEGDVR	1028	1	369	800	DFALT	5	10488	6
SEGPROC	12676	10	905	1000	24000	19	41828	29
SORT	2976	1	1	800	15000	14	36026	15
SPL	40628	30	3290	2500	32767	16	36124	46
SPOOK	7404	3	1791	800	30000	17		20
SYSDUMP	16092	5	3425	1000	16000	22		27
	626884	299						

FIGURE B-1. Listing of HP software showing the size of the program in words and the number of segments in the program file. The size of the initial stack is shown along with the maximum size it may grow to. SL segs refers to the number of SL segments that are directly referenced by the program file. These segments may in turn reference other segments. The sum of the length of all directly referenced SL segments is included under the heading SL words. Total segments is simply the sum of the program segments and the SL segments.

SYSTEM DISC LAYOUT

USE OF DISC SPACE	APPROX SECTORS	
Disc label	1	
Defective tracks table	4	
Cold load information	22	
Free space table	32	
System directory	384-6000	(configurable)
Virtual memory swapping area	1024-32767	(configurable)
System files and tables	7500	
User file area	149,000 - 187,000	(7920)

System software uses about 15,000 sectors of the user area.

On non-system discs (other than private volumes) the only overhead is for the disc label, the defective tacks table and the free space table. All other space is available for user files. Master volumes of private volume sets will have a file directory using 1000 - 4000 sectors.

Drive type	Tracks	Sectors	Bytes
7906	1,600	76,800	19,660,800
7920	4,075	195,600	50,073,600
7925	7,335	469,440	120,176,640

PUB.SYS DISC SPACE

FILE	SECTORS	FILE	SECTORS	FILE	SECTORS
APL	1201	FORTRAN	384	QUERY	383
BASIC	349	FREE2	43	RESTORE	40
BASICOMP	281	INITIAL	400	RJE	61
CICAT	2785	LISTDIR2	84	RPG	472
COBOL	718	LISTEQ2	32	SEGDVR	16
COMMAND	501	MERGE	29	SEGPROC	118
DPAN2	297	MPMON	73	SL	5001
EDITOR	284	MRJE	195	SORT	30
FCOPY	174	MRJEMON	24	SPL	362
FORMAINT	76	MRJEOUT	40	SPOOK	79
				SYSDUMP	162

DISC I/O CONSIDERATIONS

All HP7900 family drives have a 937.5 kb/sec data transfer rate and all have identical seek times of 5 ms track-track, 25 ms average random and 45 ms typical full stroke. The 7906 and 7920 each have 48 sectors/track with a 8.3 ms average rotational delay. The 7925 has 64 sectors/track with an average rotational delay of 11.1 ms.

On the average, the HP3000 is capable of completing about 30 disc transfers per second. If a program has exclusive use of the system and is reading sequential sectors without buffering, as many as 58 transfers per second may be achieved. A program randomly writing sector size blocks to a large (115,000 sector) file might see a transfer rate on the order of 20-24 transfers per second. Swapping activity must be considered since it will use some of the 30 transfers available each second.

When a user reads a record, the data will be returned in about 5 milliseconds if the logical record is already in a buffer in memory.

TERMINAL I/O CONSIDERATIONS

Every character passing to or from a terminal connected to the 3000 through the ATC (asynchronous terminal controller) causes a CPU interrupt. This is true whether the terminal is strapped for character, line or block mode. This can cause performance problems when the aggregate character rate approaches 2000 per second. While this indicates only 8 terminals can be simultaneously transferring a constant 240 characters per second, this is in fact very difficult to achieve for more than one or two seconds at a time.

PERFORMANCE IMPROVEMENT CHECKLIST	APPENDIX E

1. Reduce unnecessary logons.

2. Allocate often used program files.

3. Keep segment sizes under 5,000 words.
   Are any process stacks larger than necessary?

4. Check for file or database locking conflicts.

5. Will primary paths help database access?
   Are their long sort chains?
   Are there long synonym chains?
   Are any master data-sets more than 80% filled?
   Are all IMAGE DBCBs as small as practical?
   Use "*" in Image item lists whenever possible.

6. On-line program development team should use the textfile-masterfile' technique to reduce Text and Keep overhead.

7. Maintain sufficient free disc space.
   Is there at least 15,000 free sectors on the system disc?
   Is at least 10% of the total disc space free?

8. Sorts invoked from inside programs may run slower because less stack space is available for workspace. The stack may be left much larger than necessary for the rest of the program execution time.

9. Data files with high access rates should be evenly distributed over available drives. Program files with high swaps rates should reside on fast drives.

10. Keep system tables reasonably configured.

13. Are too many batch jobs executing concurrently?

14. Do any processes have more files open than necessary?

15. Are KSAM file blocking factors optimum?

16. Are any programs making excessive use of DEL edits?