PROJECT MAC May 9, 1973

Computer Systems Research Division Request for Comments No. 17

PERFORMANCE COMPARISON OF SYSTEMS 17.11 and 18.9

by Douglas H. Hunt

The experiments described here have been inspired by Multics users' comments that performance has degraded sometime after the release of system 18.0. These are the first four of a series of experiments designed to locate any sources of significant performance loss.

The four experiments described here have been motivated by reports from Multics users of a deterioration in system performance in recent months. Since the frequency of comments about performance increased noticeably sometime after system 18.0 was installed, a "natural" hypotheses has been to suspect that the functional additions to system 18.0 -- mostly changes in directory control -- may be contributing to a performance loss. Accordingly, these experiments have been designed to compare performance of pre-18.0 systems and post-18.0 systems. The systems selected for analysis were 17.11 and 18.9.

The purpose of the experiments is to discover (1) has system performance regressed significantly from 17.11 to 18.9, and (2) if so, what are the causes of the performance change? In particular, have the directory control modifications caused unexpected degradation? It should be noted that some decrease in performance was expected, in certain areas, as a result of the 18.0 changes. On the other hand, those modifications have definitely improved performance in other areas of the system.

Several kinds of experiments are required to answer the questions posed. To answer the first question, one wants to express system performance quantitatively, perhaps as a single number, to form a basis for comparison. To answer the second question, specific controlled environments must be set up in order to focus upon particular parts of the system.

The objective of the first experiment was to establish a single simple measure by which any two releases of the Multics system could be compared. To provide a basis for comparison, a

repeatable test load in the form of absentee scripts--with "sleep" time between commands to simulate user think time--was chosen. In each experimental trial, six absentee processes were submitted; each process issued a random permutation of a set of sixteen commands. Upon completing the sixteen commands, each absentee processes recorded its total cpu time, and continued executing, in order to maintain the total system load, until all six processes had finished their first sixteen commands. The average value of the total cpu time for the six processes served as the performance measure. The configuration chosen for this experiment was a small service system (1 cpu and 256 K), which includes a paging drum. In repeated trials, the performance measure for the two systems was stable enough that a significant difference was apparent, as shown below.

system	17.11	18.9
trial		
1	128 sec.	212 sec.
2	137 sec.	191 sec.
3	131 sec.	194 sec.
4	137 sec.	---

These data indicate that, under the experimental conditions, there was about a 50% degradation in the chosen performance measure in system 18.9. Each trial of the experiment caused about 65,000 page faults under system 17.11, and about 105,000 page faults under 18.9. At 8 milliseconds per page fault, the difference of 40,000 page faults accounts for 320 seconds, or about 55 seconds per process. The difference in performance might therefore be explained by extra paging. Based upon meter_gate measurements made on both systems, however, there did

not appear to be any changes in the performance of hcs_ entries large enough to explain such a marked difference in the overall measure.

A hypothesis consistent with these observations was formed: certain system primitives might require a larger working set in 18.9 than in 17.11, causing a larger number of pages to become "effectively wired". This in turn would cause increased user paging traffic.

The second experiment was performed in order to examine, at least indirectly, the working sets of certain directory control primitives. The following commands were issued interactively on both systems:

qedx	(create a segment "foo")
rename	(rename "foo" to "bar")
setacl	(modify the acl on "bar")
delete	(delete the segment "bar")

In addition, the "flush" command was invoked before each of the commands above, in order to flush the paging area. To eliminate interference effects, only one process was run in this experiment (the initializer process remained in the blocked state, since accounting was disabled). The configuration was also simplified: these experiments were run on the development system, since less main memory was required, and no paging device was used--eliminating pre-paging and read-write sequences. Finally, the first invocation of each command in this experiment was not included in the results. The data shown in the chart below are the demand page fault and cpu time figures obtained from the

ready message.

system command	17.11		18.9	
qedx	79	1.2	81	1.2
	64	1.1	68	1.0
	67	1.1	68	1.0
			66	1.0
rename	25	0.36	22	0.36
	25	0.36	21	0.35
	23	0.35		
setacl	36	0.48	33	0.49
	36	0.49	37	0.51
	33	0.47		
delete	42	0.53	41	0.53
	41	0.52	42	0.54
	39	0.51		

These data indicate that the working set behavior of the four selected commands, as measured by the ready message, is indistinguishable. It is therefore most unlikely that the paging behavior of a component procedure of any of these commands has changed drastically from 17.11 to 18.9. Thus, the hypothesis posed above appears to be incorrect. Since experiment one exhibited a gross performance degradation but experiment two exhibited no apparent change, each of the differences in the experimental environment needed to be considered as a possible explanation for the observed degradation. Experiments one and two made use of different commands, and only in experiment one were initial invocations of commands included in the data. In addition, experiment one was performed with six absentee processes on a small service configuration with page multi-level, whereas experiment two was performed using one interactive

process on a development system with no paging device. Concentrating on the commands issued in the two experiments, the absout files from experiment one yielded one useful statistic--PL/1 compilations appeared to generate noticeably more page faults in 18.9. Little other information comparing performance of individual commands could be gleaned from these absout files, due to the noise in the ready message data.

The third experiment was performed in an attempt to strike a middle ground between the first two; both in order to reduce noise in the results, and in order to reduce the number of variables under consideration while (hopefully) not eliminating the cause of the measured degradation. As in experiment two, the configuration chosen was a development system with no paging device, and all measurements were performed on one process, to eliminate the effects of paging interference. However, the test load was the same absentee process used in experiment one. The relative performance of each command in the script was more apparent in this experiment than in the first one. As anticipated, the PL/1 compilations were paging much more heavily under system 18.9. It was more surprising to note that the compilations accounted for practically all of the extra paging traffic observed in system 18.9. The chart below contains the demand page fault counts from the ready message for both PL/1 compilations in the script, and the total page faults for the

remaining 14 commands-- shown as the "other" category.

system	17.11	17.11	18.9	18.9
command				
pl1 list	6622	6597	18205	18316
pl1 dprint	2915	2737	10465	9980
other	5375	5334	6452	5878
total	14912	14668	35122	34174

There is thus an approximate threefold increase in the number of page faults caused by PL/1 compilations. The "other" category generates 10 to 20 percent more page faults in system 18.9 than in system 17.11.

Since the installed PL/1 compiler was version 1 in system 17.11 and version 2 in system 18.9, it became necessary to determine what portion of the degradation was due to the new compiler and what portion was due to the new system. Consequently, experiment four was planned to measure the paging behavior of both compilers on both systems. This experiment was performed on the same configuration used in experiment two. As before, all measurements were obtained from one interactive process. Each experimental trial consisted of compiling the dprint program, in both version 1 and version 2 PL/1, after having flushed the paging area. Since by the results of experiment three it is apparent that both initial and subsequent invocations of the compiler caused more paging in 18.9 than in 17.11, the simpler case of the subsequent invocation was selected

for further measurement in experiment four. The demand page fault figures from the ready messages are shown below.

system	17.11	18.9	18.11
PL/1 version			
one	1921	2452	2330
	1849	2443	2353
	1895	2353	2324
two	6013	6785	6613
	5903	6577	6530
	6010	6746	6572

Measurements from system 18.11 were included since certain changes in directory control, made to improve performance, were included in system 18.11. There does appear to be a slight improvement over system 18.9. Nonetheless, in the post-18.0 systems, the version 1 compiler caused about 30% more page faults, and the version 2 compiler caused about 10% more. The absolute increase in the number of page faults is about the same for both versions of the compiler. These data are not in conflict with the "dprint pl1" entries in the results for experiment three, since those entries indicate demand page faults for initial invocations of the compiler.

These results indicate that there are at least two components contributing to the increased paging observed in 18.9 (and 18.11) -- a component due to the new compiler and a component due to the new system. The component due to version 2 PL/1 may be attributable to decreased locality in referencing data structures. Further experiments with both versions of the compiler, made on a very lightly loaded service system (2 cpu's, 384 K), reveal that both require about the same number of page

faults -- roughly 300 -- to compile the dprint program. These results, combined with those above, imply that there is less locality in the page reference pattern of version 2 PL/1. The component of performance degradation due to the transition from system 17.11 to system 18.9 is still unexplained. Further experimentation is required to locate the cause (or causes) of this observed effect.

Thus there is, in answer to the first question posed, a decrease in performance from system 17.11 to system 18.9. At this point, one ingredient in the performance degradation has been identified -- the paging behavior of the version two PL/1 compiler. Although the installation of the version two compiler is independent of the system changes made between 17.11 and 18.9, the new compiler was part of the user interface at the time that system 18.9 was installed, so it must be considered when comparing the performance of the system at two points in time. The remaining portion of the performance loss is the subject of continuing investigation. It is still not certain whether the directory control changes contribute significantly to the observed differences, but, based upon experiment two, this appears unlikely.