MPL-55

TO: Multics Performance Log

FROM: J. H. Saltzer

DATE: June 16, 1971

SUBJECT: Measurements of Memory Interference in the one-CPU system

A series of experiments were undertaken to measure the amount of interference for access to core memory experienced by a Multics user. Two sources of requests for core memory access can be expected to significantly interfere with a CPU's access to memory: the high-performance paging ("firehose") drum and the "other" control processor, when 2 CPU's are in use. Other I/O devices request memory access infrequently relative to the CPU's and drum, so our primary interest is centered on those two devices.

The measuring tool used in these experiments was a short machine language program with known, controllable memory accessing characteristics. The nominal running time of this program, as predicted from the specified instruction timings, was in the range of 50 to 350 microseconds, depending on the initial value of a loop counter contained in the program. For a given loop count, the program was executed many times, with calendar clock readings at the beginning and end of each trial, to obtain a distribution of actual running times. The running times were divided by the number of instructions actually executed, to provide an instruction processing rate independent of the loop length. Ideally, if there were no interference effects or interrupts, all attempts to execute the program would result in about the same measured instruction execution rate.*

In the real system, however, accesses to memory by other processors, I/O devices, and the paging drum will all slow down the test program, resulting in at least some trials with an instruction rate slower than usual. When the result of many trials is plotted in a graph of frequency of observation versus instruction processing rate, the effects of interference caused by other users operating the drum and other CPU are easily identifiable.

One problem with this method is that the amount of interference experienced will depend not only on what the other users are doing, but also on the nature of the test program being executed. Construction of a truly representative Multics program is probably impossible, so the lesser goal of a program sufficiently representative to give meaningful interference measurements was tackled instead. Since interference is probably a function

---

* One would expect to find some residual variation resulting from hardware gate response times not being perfectly uniform. We presume that this effect is small compared with the effects being measured.

primarily of the rate at which instructions and data are retrieved from memory, this lesser goal reduced to construction of a program whose memory accessing rates were similar to those observed for Multics as a whole. MPL-51 reports on a series of hardware measurements made in June, 1970, of the fully loaded Multics system. From that data, one draws two important numbers: the observed average instruction rate of the CPU was 342,000 instructions per second, and the average memory access rate was 419,000 instructions per second. Thus, for a benchmark, one can construct a program which has approximately those properties. After some experimentation, which involved selective insertion of indirect references and double precision load instructions, the following eleven instruction sequences appeared to have average behavior approximating that reported in MPL-51:

```
eaxl      loopcount
lda       y
aos       w
ada       x
sbaq      z
ada       y
eapbp     bp|0,*
ldaq      bp|0
eax0      -5
eaxl      -1,1
tnz       -9,ic
```

This sequence, which includes a loop to repeat the last ten instructions ten times, was surrounded by a pair of read clock instruction. The expected execution time of the sequence, for a loop count of ten, is about 290μsec. To insure repeatability, register bp was loaded with the address of an ITS pair which pointed to itself, before the sequence above was entered.

Finally, the program described above was implemented as an impure procedure stored in a single page, so that all data and instruction references for a single trial would be directed to the same memory box. A PL/I program was constructed to call the test program n times in succession, go to sleep for 10 seconds (to allow the test program to be paged out) and then call it again n times. This outer loop was repeated m times. Typical values for n and m were 50 and 100, respectively, thus obtaining 5000 samples over a period of 15 to 30 minutes. Samples which measured more than 1000μsec. were discarded, on the assumption that they represent sequences which were interrupted. To display the results, the 5000 measurements of instruction processing rate were divided into equal-sized bins representing about a 1% separation at the mean value and the number of samples in each bin were printed out. Figures one and two are typical examples of the display.

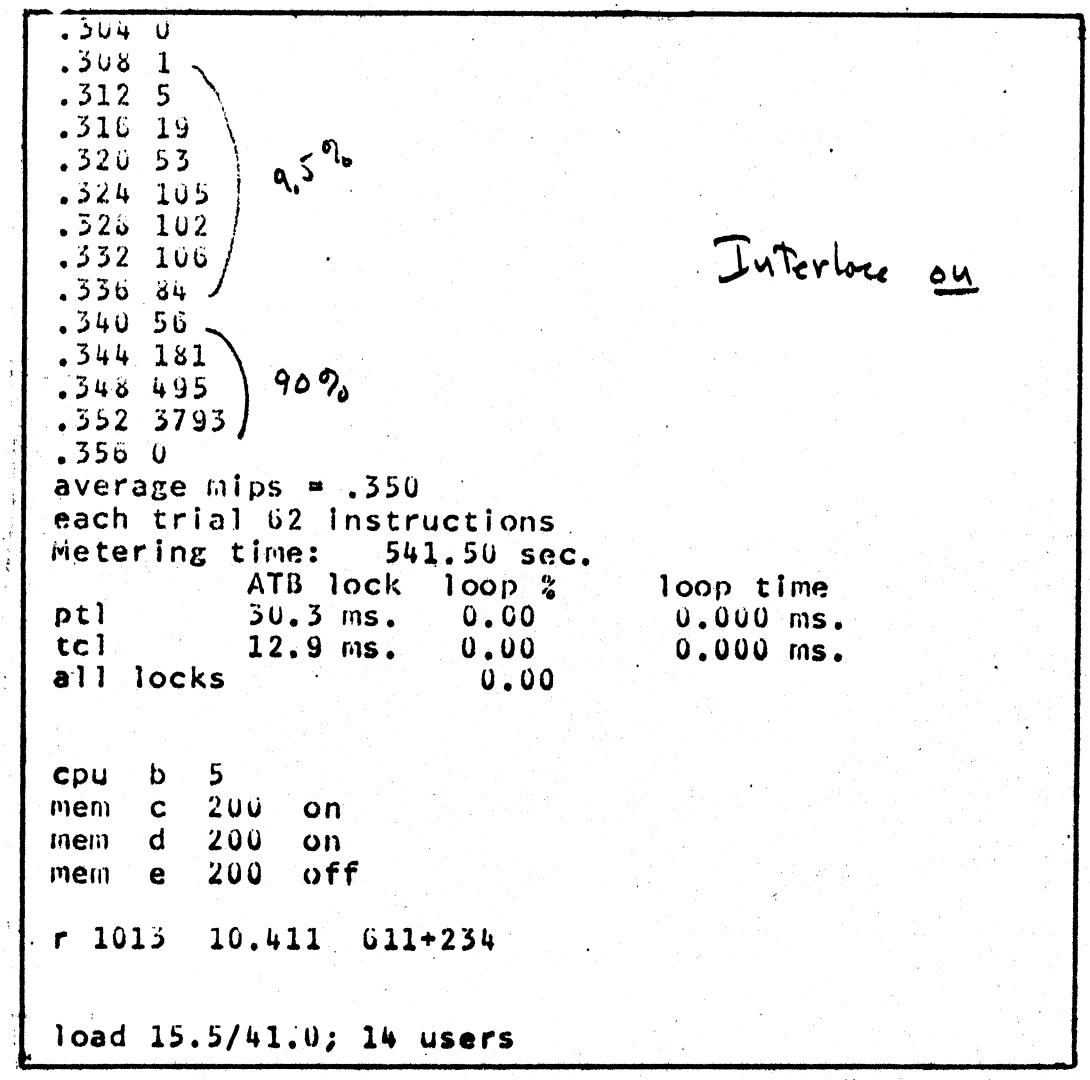

Figure one -- distribution of CPU execution speed with interlace in operation

```
.220  0
.224  1
.228  0
.232  1
.236  3
.240  5
.244  13
.248  15
.252  41
.256  59
.260  32
.264  88
.268  22
.272  49
.276  1
.280  0
.284  5
.288  9
.292  24
.296  47
.300  43
.304  37
.308  2
.312  2
.316  8
.320  6
.324  6
.328  14
.332  16
.336  36
.340  136
.344  496
.348  1746
.352  2037
.356  0
average mips = .343
each trial 62 instructions
Metering time:   995.86 sec.
          ATB lock  loop %      loop time
ptl        10.3 ms.   0.00        0.000 ms.
tcl         5.0 ms.   0.00        0.000 ms.
all locks             0.00


cpu  b  5
mem  c  200  on
mem  e  200  on
mem  d  200  off


Multics 15.8 up since 05/18/71  2120.7
load 35.5/41.0; 36 users
```

6.4 %

3.5 %

90 %

Interlace off

← configuration

Figure two -- distribution of CPU execution speed without interlace

One-CPU measurements

Figures one and two also serve to illustrate the effect of drum interference when one CPU is in operation, as well as the effect of memory interlace in reducing that interference. In both figures, a majority of the samples appear at .352 million instructions per second (mips). However, in figure one, a secondary clustering of samples appear near .328 mips; about 10% of all the samples are in this secondary cluster. One may presume that these samples were taken at times when the firehose drum was in operation. The drum will, with interlace on, affect every trial when it is operating, so we conclude that the drum was operating about 10% of the time. The page table lock interval meter indicates a value about three times as large as it does when the system is fully loaded (see figure two for an example), so that a fully loaded system should produce a secondary peak containing 30% of the samples. The average instruction execution rate with the drum in operation is about 7% slower than with the drum off; if this condition holds 30% of the time when the system is fully loaded, the average slowdown is 30% x 7% = 2%.

Figure two is the corresponding result without interlace. This time, two secondary peaks are observed. The number of samples enclosed by the two peaks taken together is about 10% of the total, and again probably represents drum interference. However, with interlace off, the drum will not always access the same memory controller as does the CPU. If we assume pageable core is divided among the two controllers with 50 pages in one and 100 pages in the other, the drum and CPU would be accessing the same controller about 55% of the time. Thus we conclude that the drum was in operation about $\frac{10\%}{55}$ = 18% of the time during this experiment. Explanations of the two peaks are harder to come by. One hypothesis is that since the firehose drum requires a 2μsec. double precision memory cycle every 4μsec., the CPU synchronizes with it, in one of two modes. The lower peak, at .256 mips, corresponds to execution of exactly one CPU instruction every 4 microseconds, which strongly suggests synchronization with the drum. The other secondary peak, at .296 mips, perhaps represents trials in which starting conditions either did not lead to synchronization of the drum and CPU, or synchronization in some other mode occurred. (The CPU habit of retrieving instructions in pairs leads to an access time pattern which could synchronize in at least two ways.)

In any case, the average loss due to interference when interlace is off was observed to be about 3%; if the drum was operating 30% of the time this figure would climb to 5%. With interlace on we expect a loss of about 2%. The difference in these two figures, about 3%, may be taken to be the value of interlace in the one-CPU system -- about enough capacity for 2 more users.

Before banking too heavily on this result, it is worth noting that one would expect that turning interlace off should cause about twice as large a slowdown, but only half as often in a 2 memory system, and differences should be second order. Since our measurement showed that the slowdown was substantially more than twice (about a factor of four) it suggests that detailed synchronization effects are very important in determining overall performance. The implications for potential model building are severe -- a prediction of the performance of an uninterlaced system may require a model which contains much detail.

Further measurements, using the same technique, on the 2-CPU system have been started, and will be reported in a later MPL.