Massachusetts Institute of Technology
Programming Development Office
Cambridge, Massachusetts 02139

RECEIVED
APR 9 1975
J. H. SALTZER

To: Robert C. Daley

From: Gary C. Dixon

Subject: Running MACSYMA under Multics

Date: March 28, 1975

Copies: Corbató
Dertouzos
Moses
Schroeder
Clark
Moon
Reed
Redell

This memo presents the results of a comparison of two implementations of the MACSYMA System (Project MAC's SYmbolic MAnipulation System): the implementation supported by the MathLab Group of Project MAC on a Digital Equipment Corporation PDP-10 Model KA-10 computer running the ITS Time-Sharing System; and the implementation supported by the CSR Group of Project MAC on a Honeywell Model 6180 computer running the Multics Operating System. These implementations are hereinafter referred to as the ITS implementation and the Multics implementation.

The goal of the comparison was to determine how large a Multics configuration would be required to support 20 MACSYMA users (only 10 of which are active at any given time), in addition to the configuration required to serve the current IPC Multics load of 60 users. The comparison was only a feasibility study. No attempt was made to determine which implementation of MACSYMA is most appropriate for a MACSYMA service, nor was there any determination of what price the "average" MACSYMA user would have to pay on either system.

The starting point for the comparison was the ITS implementation of MACSYMA, which currently supports 10 MACSYMA users (only 5 of which are active at any given time) with acceptable response times. The performance of ITS MACSYMA was compared with that of Multics MACSYMA, and the results were extrapolated from 10 up to 20 MACSYMA users. It was assumed that the majority of the MACSYMA users would log in to Multics via the ARPA Network.

Before describing the experiment which was performed, a brief description of the hardware and software for the two implementations is in order.

THE HARDWARE

The ITS implementation runs on a DEC PDP-10 Model KA-10, to which special paging hardware has been added. The memory for the system is 512K (36-bit) words of 1.8-usec memory.

The Multics implementation runs on two Honeywell 6180 processors, each of which includes paging hardware, associative

memory registers, and a cache with 120-nsec or less access time. The 2-level memory for the system includes 384K (36-bit) words of .5-usec main memory, and 2M (36-bit) words of 410-usec bulk store.

THE SOFTWARE

The tests described below were run on both systems under MACSYMA Version 253, which has just recently been installed. The MACSYMA System itself is written in the LISP Language. The ITS and Multics systems both provide LISP compilers which were used to generate executable object code for the two MACSYMA implementations.

THE EXPERIMENT

The experiment consisted of executing a series of 27 demonstration files under both implementations of MACSYMA. The demonstrations cover a broad spectrum of MACSYMA functional capabilities, including: solving systems of algebraic equations; numeric evaluation of differential equations; conversion of laplacian equations from cylindrical to cartesian and spherical coordinates; finding the greatest common denominator of two equations; factoring equations; plotting functions; taking the limit of expressions; manipulating and solving matrices; solving second order linear differential equations; and many others.

From a Multics process, the files were transmitted through an ARPANet connection to three processes running MACSYMA on each of the two systems. Preceding and following each of the demonstration files was a

```
?runtime();
```

request which invokes the LISP runtime function from MACSYMA. This function returns the total runtime of the process running MACSYMA in CPU seconds. The runtime before was subtracted from the runtime after to obtain the total CPU time required to execute each demonstration.

THE RESULTS

Figure 1 shows the results of the runtime measurements for the ITS implementation and for the Multics implementation. The ratio of runtimes on Multics versus ITS is also shown. The TOTAL line shows the cumulative runtime for all of the demonstrations on each implementation.

Although the values of the LISP runtime function provided under the two implementations are not exactly equivalent, they

are nearly so and they do reflect the relative amount of processor resources required to run the two MACSYMA implementations. Both implementations exclude paging activity from the runtime measure. Both implementations exclude the time spent loading MACSYMA into the user address space. However, the ITS implementation also excludes ARPANet I/O from the measure, assigning CPU time for I/O to the command process (the father of the MACSYMA process). Measurements of command process runtime during the experiment show that the ARPANet I/O adds from 3 to 5% to the times shown for the ITS implementation in Figure 1. If this discrepancy were accurately measured and accounted for in Figure 1, it would act to reduce the Multics/ITS runtime ratio.

Figure 1
Runtimes for the MACSYMA Demonstrations
(in CPU seconds)

Demonstration	ITS	Multics	Multics/ITS
algsys.demo	23.70	15.05	63.49 %
at.demo	2.60	3.95	151.72 %
c2cyl.demo	4.18	4.40	105.41 %
common.demo	22.11	11.66	52.75 %
cyl2c.demo	2.93	2.94	100.17 %
cyl2s.demo	7.85	5.44	69.25 %
ezgcd.demo	78.90	64.42	81.64 %
factor.demo	18.64	8.16	43.77 %
fgtgen.demo	15.97	9.73	60.91 %
fgtrat.demo	9.72	7.58	78.03 %
gen.demo	18.78	18.21	96.96 %
gfacto.demo	26.28	16.55	62.97 %
graph.demo	17.36	13.66	78.69 %
laplac.demo	17.63	16.40	93.00 %
legen.demo	28.64	25.72	89.82 %
limit.demo	28.96	17.23	59.50 %
mat.demo	9.77	3.75	38.38 %
matrix.demo	8.29	5.62	67.88 %
ratsub.demo	6.94	4.81	69.25 %
risch.demo	10.40	7.70	74.10 %
simpl.demo	25.01	8.24	32.96 %
sin.demo	77.39	49.52	63.98 %
solder.demo	12.02	7.84	65.23 %
solve.demo	29.04	19.29	66.43 %
sum.demo	1.84	2.44	132.17 %
taylor.demo	5.15	5.49	106.49 %
trig.demo	30.42	27.22	89.46 %
TOTAL	540.52	383.02	70.86 %

CONCLUSIONS

Our experiment shows that running MACSYMA under Multics requires only about 70% of the CPU time used by MACSYMA under ITS.

Given this result, and given the following assumptions: (A) that the ITS implementation of MACSYMA currently serves 10 users with adequate response time, providing that no more than 5 users are active at any given time; (B) that 1.5 6180 CPUs are required to support the current Multics user load (60 users); and (C) that as the number of users increases on the two systems, the system overhead is about equal on both ITS and Multics. We can then conclude that Multics would have to dedicate the computing power of .7 6180 CPUs for each 10 MACSYMA users to be served.

The first assumption (A) was given as part of the study, and there seems to be a concensus among those who know about ITS operating characteristics that it is valid.

The second assumption (B) has been verified by recent measurements of Multics system performance, which show that the current Multics configuration can adequately handle an 80 user load. Thus:

$$2.0 \text{ CPUs} * \frac{60 \text{ current users}}{80 \text{ possible users}} = 1.5 \text{ CPUs}$$

Based upon these measurements, I am fairly confident of the validity of the second assumption.

The third assumption (C) might seem reasonable at first glance but it is difficult to prove. Obtaining measurements to support an adequate proof would require more effort than PDO can now afford to spend. However, some information on the validity of this assumption is available.

Rough measurements made on ITS with Dave Moon's smeter command indicate that system overhead on ITS running 4 MACSYMA processes (8 MACSYMA users, 1/2 running at any time) ranges from 20 to 25% of the total system CPU time. Similar measurements on the current Multics configuration show a system overhead of about 56% of the total system CPU time (1.12 6180 CPUs). However, Dave Jordan has estimated that Multics system overhead could be reduced by about 25% (.5 6180 CPUs) by replacing the 2M bulk store in the current configuration with 1.75M additional main memory. This reduction comes mainly from the lowered paging overhead with additional main memory. This reduction would bring Multics system overhead in line with ITS system overhead. Thus, meeting the third assumption requires a Multics configuration of 2 CPUs and 2M memory with no bulk store.

OBSERVATIONS

A Multics configuration consisting of 2 CPUs, 2M words of main memory, and no bulk store could easily support 10 new MACSYMA users plus the present Multics user community. As stated above, the reconfiguration to 2M memory shows that the system paging overhead would be reduced by about .5 6180 CPUs. Thus, we have:

(1.5	-	.5)	+	.7	=	1.7
current load		gain from 2M memory		10 MACSYMA users		total CPU used

(all numbers are fractions of 6180 CPUs)

This calculation shows that, when running the current IPC user load plus 10 MACSYMA users on the 2 CPU, 2M memory system, there would be excess CPU capacity of .3 6180 CPUs. With this slack capacity, it is safe to assume that the reconfigured Multics could support a trial MACSYMA service for 10 users. By using this excess capacity, we could support up to 14 MACSYMA users. However, a third CPU would be needed to provide a full 20 user MACSYMA service. Such a 3 CPU, 2M word memory system should support up to 25 or 30 MACSYMA users.

Determination of the DEC-10 configuration needed to support 20 MACSYMA users was outside the stated goals of this study. However, a recent presentation and conversations with DEC representatives provided performance information for DEC's new Model KL-10, relative to the KA-10 used in the above experiment. Extrapolation of the measured performance of a KA-10 running MACSYMA onto the new KL-10 indicates that a KL-10 with 512K words of .9-usec memory could support 20 MACSYMA users, and by increasing memory size to 2M words, could support 40 or more MACSYMA users.