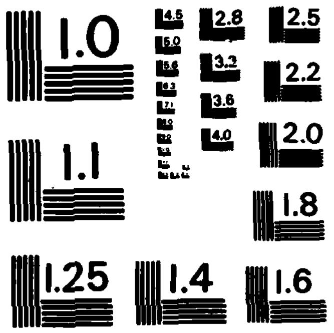

1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Defence and Civil Institute of Environmental Medicine

3

June 1982 DCIEM Research Paper 82-P-23

A UNIX Benchmarking Tool
with Results from the PDP-11/44,
VAX-11/780, and
Perkin-Elmer 3242

Martin Tuori

Defence and Civil Institute of Environmental Medicine
1133 Sheppard Avenue West, P.O. Box 2000
Downsview, Ontario M3M 3B9

DEPARTMENT OF NATIONAL DEFENCE – CANADA

A UNIX Benchmarking Tool

with Results from the PDP-11/44, VAX-11/780
and Perkin-Elmer 3242

Martin Tuori

D.C.I.E.M.
PO Box 2000, Downsview,
Ontario, Canada M3M 3B9
and
Computer Systems Research Group
University of Toronto
Toronto, Ontario

unix net address: decvax!hcr!dciem!martin

This is DCIEM Research Paper 82-P-23

Abstract

A facility is described that offers a high level means of exercising and performance testing a UNIX[1] timesharing system. Its structure is explained, and resulting issues and limitations are discussed. Results have been obtained on several different computer systems, including a PDP-11/44, a VAX-11/780, and a Perkin-Elmer 3242, each running a variation or extension of Version 7 UNIX.

1) Background

This facility was originally developed as a system exerciser, to check the integrity of a PDP-11 based UNIX system that had undergone hardware repair. The system had been unstable under heavy load, such as generated by the normal user community, yet it passed all hardware diagnostic tests. Rather than release the newly repaired system to the users and risk further crashes, we decided to simulate that heavy load in a more controlled context. Several typical user level processes were selected, and several copies of each were run simultaneously. The results of each process were compared byte for byte with a trusted copy prepared earlier. This proved to be an excellent barometer of system health, and has been in use for about 3 years.

[1] UNIX is a trademark of Bell Laboratories
DEC, PDP, and VAX are trademarks of Digital Equipment Corporation
Perkin-Elmer is a trademark of Perkin-Elmer Corporation

By using the *time* command to measure the elapsed time required to complete a set of exercises, we obtained an overall measure of the performance of the system, as a hardware/software unit. This became useful in evaluating changes to the UNIX kernel, such as the number of system buffers, disc block interleaving (mkfs parameters), etc. Most recently, this same tool has been used to measure the throughput of UNIX systems from different vendors.

2) The Benchmarking Tools

The benchmarking tools are a collection of command files, source files and data files, the focus of which is a single command file in the topmost directory. This *run* file permits the operator to initiate multiple copies of user level processes (jobs) which are to run concurrently. Four types of processes are used -- a compile of a 729 line C program, formatting a section of the UNIX online manual using nroff, performing global substitution in a text file using the text editor *ed*, and execution of a moderate floating point process.

Each process type lives in its own directory, where, in addition to the necessary source and data files, there is a run file that support three operations – setup, run, and cleanup. The purpose of the setup phase is to run the process once, to obtain a *true* version of whatever output is obtained, and to create a sub-directory for each instance of the process that is to be run. The run phase forks the desired number of instances, and awaits their completion. The cleanup phase cleans out the sub-directories, etc.

The topmost *run* command mirrors the setup-run-cleanup structure of each process type by providing this choice of actions to the user, either as arguments or through an interactive dialogue. It also provides an action to select the numbers of each process to run, as well as the options of checking the validity of the outputs produced (by using the *cmp* command against the *true* version), and of noting the time taken to run the batch. This structure results in one shell for each user level job, plus whatever children processes are required, and cmp, expr, and test processes.

Several issues arise in considering the significance of these tests; these are addressed in the following paragraphs. The *check* option is a comparison against a local standard, assumed to be correct, not against a universal standard for all UNIX systems. While such a universal standard might be appropriate for the work done by the text editor, it is certainly not useful for the C compiler, since the target machine is not constant. The other process types fall somewhere in the middle; a slight change in the spacing produced by nroff might be acceptable to most users, yet would fail in a simple byte by byte comparison with a universal standard.

The UNIX *time* command produces data with some variance, but this is minor at moderate load.. For example, 7 trials of the benchmark were run at 60 processes on the P-E, giving the following average times:[2]

real: 15:43 user: 11:13 sys: 3:51

with standard deviations of 3.3, 0.6, and 2.6 seconds respectively.

Version 7 UNIX and its variations permit each user a limited number of processes at any one time. To overcome this limitation, it is necessary to run the benchmark as the superuser. This uncovers an ancient bug in the *cc* command (C compiler), which remains in some systems. The method used for creating names for temporary files is adequate for multiple compiles run by normal users. When run by the superuser, however, multiple compiles collide, resulting in errors. The correction to this is to use the library routine *mktemp* to create the names for the temporary files. It was also necessary to remove the optimization option from the C compile process, since the UNIX for the P-E does not yet have an optimizer for the C compiler. Requesting it causes no action, which would favour that system unfairly. The P-E system does, however, suffer the disadvantage that its utilities are unoptimized, at this time.

This is typical of the benchmark's ability to uncover problems in UNIX systems that normal user communities do not encounter. As the load is increased, various resource limits are reached within the kernel, often causing the system to crash. Such problems are sometimes remedied by recompiling the kernel with more of the resource(s) in short supply. One problem that requires a different type of change is a bug associated with the element *f_count* in file.h; it is defined as a char in Version 7. Since all the processes share an open file (/dev/tty), this count overflows, causing (we believe) the file structure to be reassigned. Pandemonium results. While this does not arise under normal conditions, the count has been changed to a short integer to correct the problem.

Finally, The benchmark does not make any provision for *think time*, during which a user's processes will be inactive. Nor does the benchmark exercise terminal I/O in proportion with the processing that is carried out; only the controlling terminal is active, and traffic there is light. In its current form, the benchmark does not attempt to simulate a quantifiable number of users; it only loads the system as fully as possible. Similarly, no effort is made to exercise peripheral devices other than the discs used for files and swapping.[3]

---

[2] The *time* command reports the total elapsed time (real), the time spent by the CPU on user code (user), and the time spent by the CPU on system code (sys). These times are given as minutes:seconds.fractional seconds.

[3] A separate tool, called *devex*, was designed to exercise peripherals simultaneously. It has been used with a magtape drive, frame buffer, and film recorder; one copy of it may be run concurrently with the benchmark.

3) Specific Results

For purposes of comparison, the benchmark has been run on three different systems, configured as follows:

- PDP-11/44, 256kb memory, floating point processor, one 80mb disc, running Version 7 UNIX.
- VAX-11/780, 3mb memory,[4] floating point processor, 2 disc controllers, 3 300mb discs, running 4.1BSD.
- Perkin-Elmer 3242, 3mb memory, floating point processor, 2 disc controllers, 3 300mb discs, running The Wollongong Group's Edition VII UNIX.

The two larger systems have very similar configurations; the root filesystem is on a controller and disc by itself, /tmp and the user filesystem are on separate discs on the other controller.

The four types of processes involved are of short duration; individually, they take the following times (in sec.) on the P-E:

editor:	real	10.2	user	4.4	sys	3.8
C compile:	real	39.0	user	26.5	sys	7.1
floating point:	real	16.0	user	12.7	sys	2.2
nroff:	real	9.6	user	5.0	sys	3.1

and on the VAX:

editor:	real	11.6	user	3.7	sys	2.4
C compile:	real	30.5	user	20.3	sys	3.5
floating point:	real	25.0	user	23.5	sys	1.3
nroff:	real	9.0	user	4.4	sys	1.9

The first graph shows the elapsed time for each system, as reported by the *time* command, as a function of the number of processes running. Since it is difficult to see variations in the slope of these curves, we have plotted the total times divided by the number of processes. The next three graphs show *user time*, *system time*, and *elapsed time* as a function of the number of processes. We can see that user time is fairly constant with load; this is expected, since the user programs have a fixed number of instructions to be performed. Those instructions may be spread out over a long period, but the work done (time taken) for each process remains the same. The Perkin-Elmer system is the fastest in this comparison, due entirely to the factor of two on

[4] The system has a total of 6mb of main memory; CSRG was kind enough to disable half of it for these tests.

the floating point process. The system times grow slowly with load. This probably indicates an increasing effort by the operating system at such tasks as scheduling, swapping, and virtual memory management. In this component, the VAX system is considerably faster, due probably to the performance tuning done by the developers of 4.1BSD UNIX. The graph for elapsed time shows that the savings in system time on the VAX are enough to make it the fastest system tested. The elapsed times drop as the number of processes grows from the lowest levels; we attribute this to processes becoming I/O bound, at which time the CPU may become idle. As the load grows, there is enough work outstanding that a system may be able to keep each disc controller and the CPU busy, thus reducing wasted idle time.

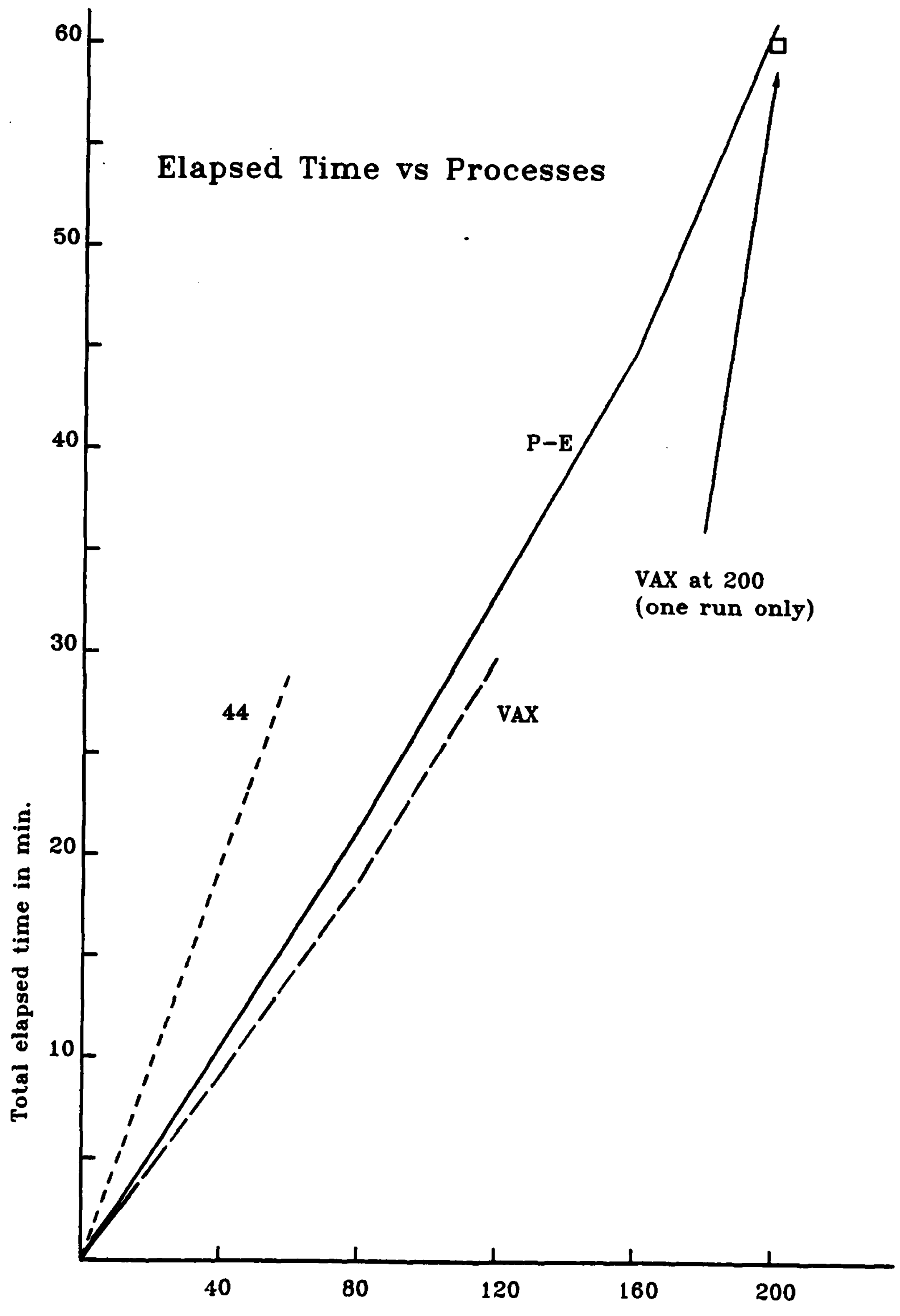
It is important to note that the two larger systems are too slow for effective timesharing when they are running more than 60 or 80 active processes. While the systems can cope with larger loads, the delays experienced at a terminal are greater than most users would accept. We were unable to load the 11/44 past 60 processes -- it simply doesn't have sufficient resources. We were able to run one test of the VAX at 200 processes, but a problem in the report from *time* forced us to treat that result separately. The elapsed time for that run is indicated as a small box on the graphs.

4) Conclusions

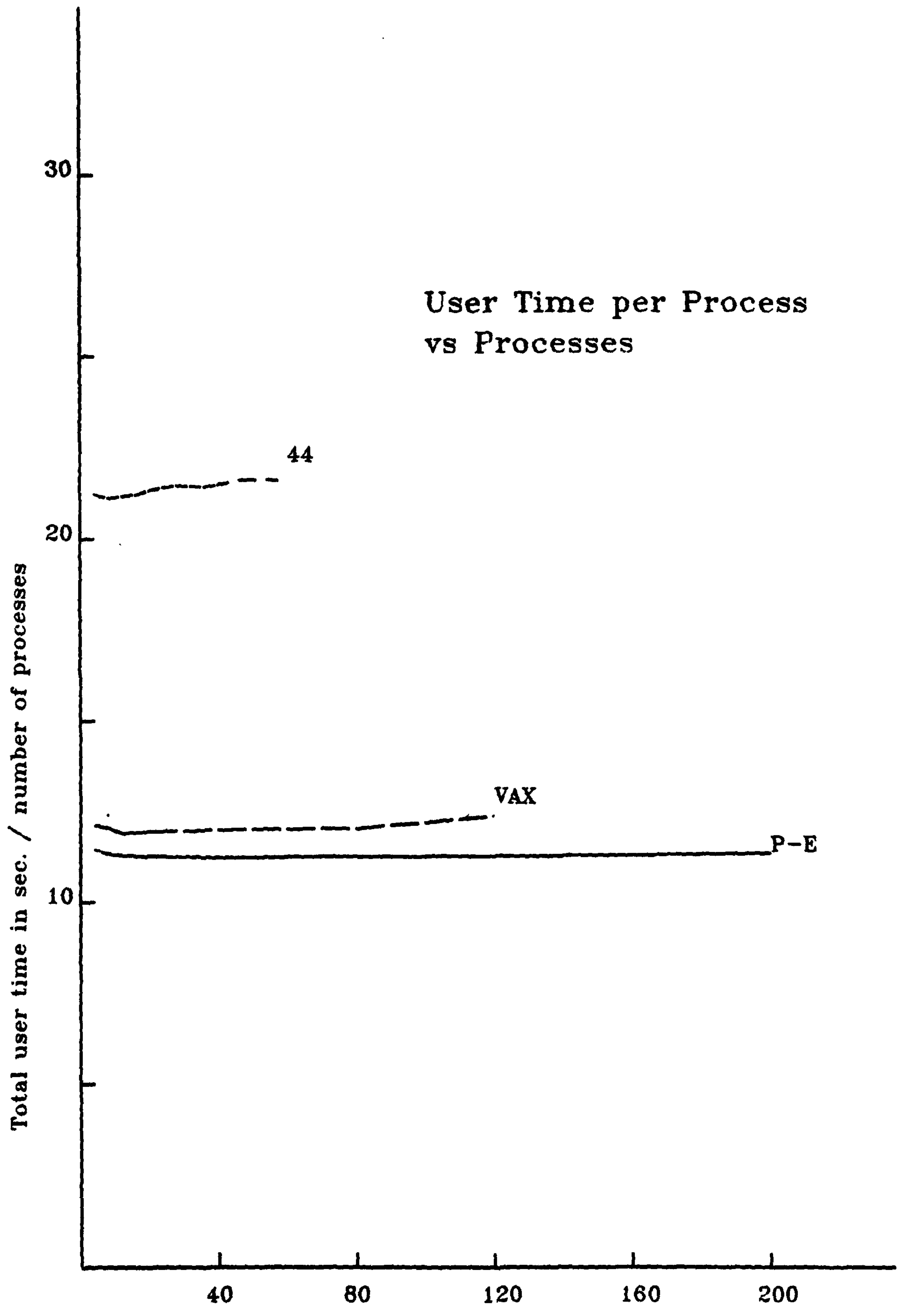
A high level benchmark technique that simulates a group of active users on a UNIX timesharing system has been described. This technique treats a system as a hardware and software package, and gives elapsed times for a blend of typical process types. It can easily be modified to reflect a different balance among the process types, or to include other processes. It would be straightforward to add think time to the benchmark, and terminal output (terminal input is more difficult, and normally a smaller percentage of overall terminal I/O). We hope that these results, and the ability to test further enhancements to UNIX systems will encourage the developers of these and other systems to evaluate and improve their products. The benchmarking package will be made available through the distribution facilities of the USENIX Association.

Acknowledgements

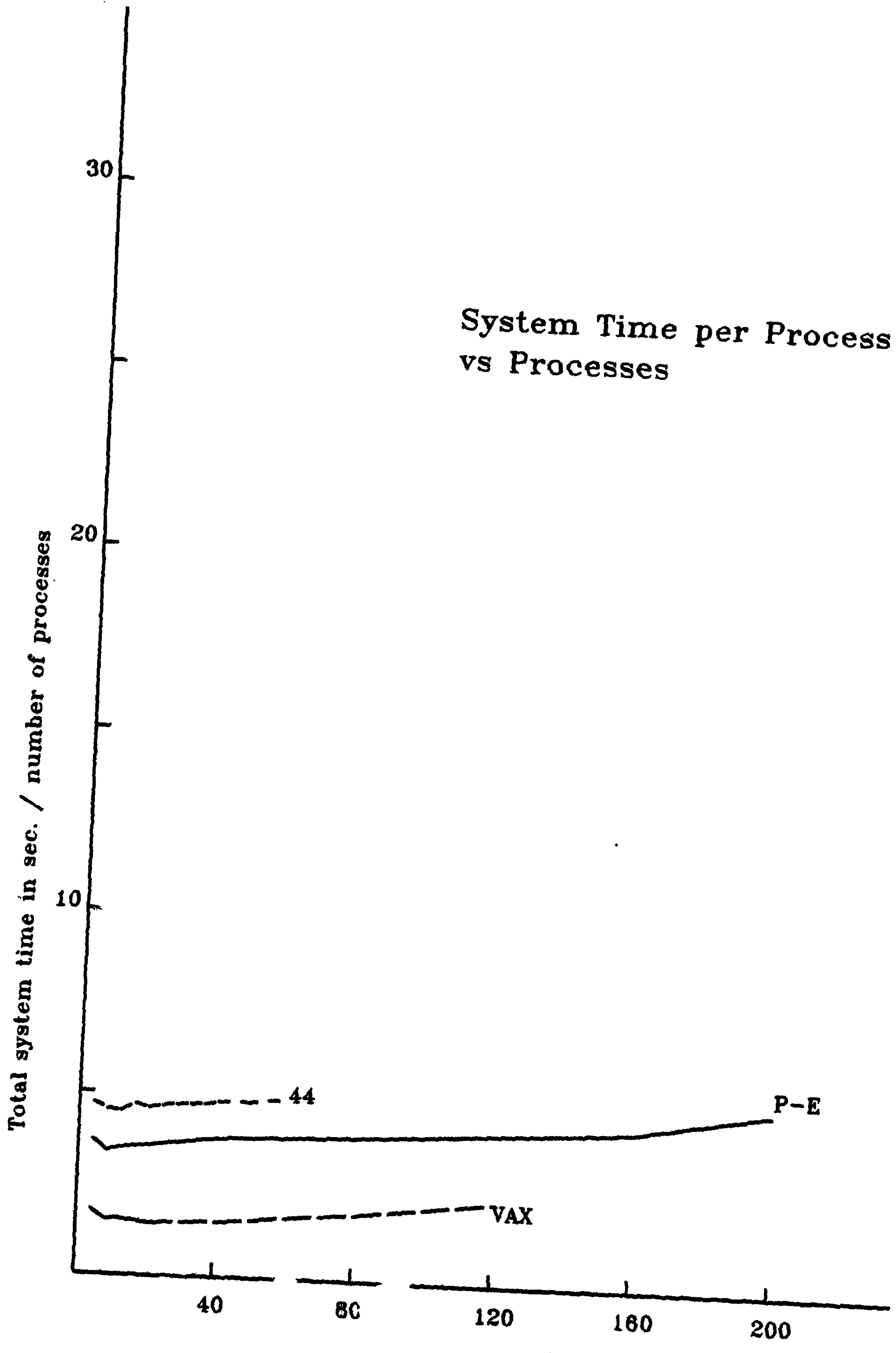
I would like to thank Dave Galloway and Dave Wortman for their encouragement, and for providing time on CSRG's VAX system to run the benchmarks. Sandra Wright coauthored the original version of this benchmarking facility.

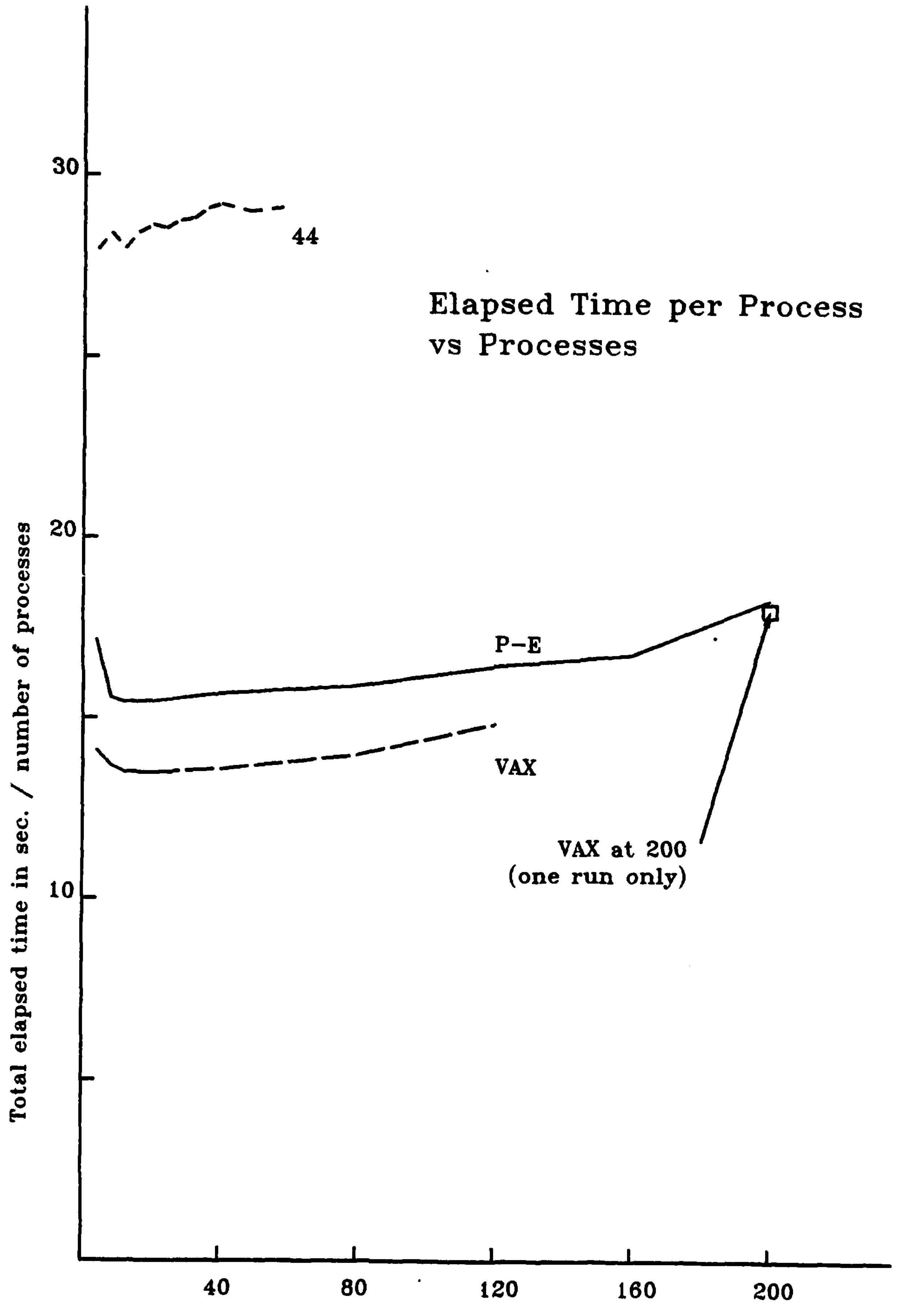


Elapsed Time vs Processes
Total elapsed time in min.
60
50
40
30
20
10
40
80
120
160
200
P-E
VAX
44
VAX at 200
(one run only)



User Time per Process
vs Processes
44
VAX
P-E
30
20
10
40
80
120
160
200
Total user time in sec. / number of processes


System Time per Process
vs Processes
Total system time in sec. / number of processes
30
20
10
44
P-E
VAX
40
80
120
160
200



Elapsed Time per Process vs Processes
Total elapsed time in sec. / number of processes
30
20
10
40
80
120
160
200
44
P-E
VAX
VAX at 200 (one run only)