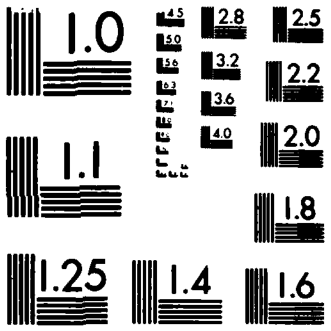
END
FILMED
DTIC


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

(2)

AD-A160 985

MRC Technical Summary Report #2840

WHAT KIND OF WORKSTATION SHOULD I BUY? SEVEN-FOLD CRITERIA FOR HARDWARE AND SOFTWARE

L. Ridgway Scott

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

July 1985

(Received May 30, 1985)

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

-A-

UNIVERSITY OF WISCONSIN-MADISON
MATHEMATICS RESEARCH CENTER

WHAT KIND OF WORKSTATION SHOULD I BUY? SEVEN-FOLD CRITERIA FOR HARDWARE AND SOFTWARE

L. Ridgway Scott[*]

Technical Summary Report #2840
July 1985

ABSTRACT

The problem of designing a workstation suitable for an individual scientist or group of scientists ~~will be~~ are addressed from an abstract point of view. A set of seven criteria concerning hardware are presented that should be considered when assessing one's needs and comparing various available products. Seven categories of software are also discussed that are critical for someone doing code development, testing and production runs on a remote computer (mainframe or supercomputer). The criteria discussed are intended to be a model checklist relevant especially for people doing code development and production work in scientific computation. After discussing criteria for evaluating workstations, two examples are given of inexpensive yet powerful systems that should be possible to assemble today.

AMS (MOS) Subject Classification: 00A99

Key Words: workstation, hardware, software

Work Unit Number 6 (Miscellaneous Topics)

[*]Department of Mathematics, University of Michigan, Ann Arbor, MI 48109-1092.

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

SIGNIFICANCE AND EXPLANATION

This is the summary of a talk given at the ARO Workshop on Microcomputers at the University of Delaware in May of 1985. It addresses the question posed in the title and attempts to give criteria to use in making such a decision.

The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the author of this report.

WHAT KIND OF WORKSTATION SHOULD I BUY? SEVEN-FOLD CRITERIA FOR HARDWARE AND SOFTWARE

L. Ridgway Scott*

People doing scientific computation are faced with a bewildering situation when they begin to purchase a workstation to help with the development and testing of codes and the interpretation of data from production runs. This technology is evolving so rapidly that choices are often not clear cut. One is constantly bombarded with claims and counter claims, and there is a real danger of designing a system using pieces of hardware or software that turn out to be "vaporware." I have been asked on a number of occasions by friends, colleagues and administrators to answer the question posed in the title, and this note presents the issues I see as being most important. The problem of designing a workstation suitable for an individual scientist or group of scientists will be addressed from a fairly abstract point of view. A set of seven criteria concerning hardware are presented that should be considered when assessing one's needs and comparing various available products. The seven criteria are an expansion of the 5M criterion of Dick Phillips of the University of Michigan, the 4M machine related to the SPICE project at Carnegie-Mellon [2] and the 3M workstation concept under development at Brown University [3]. Seven categories of software are also discussed that are critical for someone doing code development, testing and production runs on a remote computer (mainframe or supercomputer). It is especially important to consider software as an integral part of the workstation; frequently too little money is budgeted for software, and the potential value of the workstation is not realized. The criteria discussed here do not exhaust all possibilities, nor are all of them critical to everyone. They are intended to be a model checklist relevant especially for people doing code development and production work in scientific computation. After discussing criteria for evaluating workstations, I will give two examples of inexpensive yet powerful systems that should be possible to assemble today.

Hardware

There are (at least) seven variables that are important to consider regarding hardware when designing a scientific computing workstation. They are all quantitative in nature, and we shall try to weight them so that an ideal system has all variables of order unity. We are therefore prejudicing to a nontrivial degree our own preferential weights for the various categories, and individuals should not be intimidated by someone else's large need in a specific area. Also, there is no need to feel that having ANY of the variables ≥ 1 is necessary, as a small personal workstation with a good balance could be just as useful as a more powerful one that must be shared. In any case, if the budget is limited, there may be no other choice. The seven-vector we use to describe a workstation has the components $x_1, x_2, \ldots, x_7$ which indicate that the system

*Department of Mathematics, University of Michigan, Ann Arbor, MI 48109-1092.

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

executes x_1 million instructions per second,

performs x_2 million FLOPS (floating-point operations per second),

has a bus that transfers x_3 million words per second,

has x_4 million words of program/data memory,

has x_5 million bytes of (raster) graphics display memory,

has x_6 million lines of disc storage

and can communicate with its neighbors at x_7 million bits per second.

Some words of explanation are in order concerning items 5 and 6.

Raster graphics memory can be allocated in two ways: for spatial resolution or for color resolution. Using anti-aliasing techniques, these are somewhat interchangeable. For example, one could have a one-megabyte display with the following possibilities:

pixel resolution	number of colors
2K by 2K	four
1K by 1K	256
512 by 512	full color (32 bit-planes)

Graphics displays with the latter two sets of parameters are now commonly available. Certain displays are software switchable between these two options. By "full color," we mean a color resolution that is sufficient to reproduce typical photographs to an accuracy acceptable to most humans. Actually, 24 bits per pixel are considered sufficient for this purpose. The intensity variations allowed by a large number of colors permit the increase of apparent spatial accuracy by a method known as anti-aliasing. A straight line drawn close to parallel to one of the axes displays noticable "jaggies" which can be reduced by making neighboring pixels interpolate (in color space) between the colors of the background and line.

One area in which current workstation specifications are often deficient is that of graphics display resolution. It used to be very expensive to get one-megabyte displays, but they are now quite reasonably priced, as will be pointed out below. Also, graphics software has achieved an admirable level of standardization, so the inexpensive frame buffers need not be user-unfriendly. Graphics equipment can still be very expensive, but the expense is related to the speed of redrawing (e.g., in rotating a three-dimensional object). However, if speed is not critical, good resolution can be obtained on a moderate budget.

By "lines" of disc storage, we simply mean that a line should be thought of as 80 characters (or equivalently, 80 bytes). Thus $x_6 = 1$ means an 80 megabyte hard disc. The transfer rate for data to and from the hard disc is also an important consideration in many cases. It may also be closely related to the value of x_3.

Items 1-3 are obviously tightly coupled. If x_1 or x_3 is small, it will be hard for x_2 to be very large. Typical number-crunching algorithms have two arithmetic operations per data transfer (cf. the architecture of the Cray 1), so one would expect $x_2 \leq 2x_3$ in general. Array processors offer the hope of high-performance computation at low cost, but the bus-transfer (or other method of data-transfer) rate is critical. An excellent set of benchmarks has been compiled by Jack Dongarra [1] that compare x_2 for the entire gamut of computers, from PC's to supercomputers, on some computationally intensive problems.

We have left out of the criterion vector X some important hardware considerations, namely the number of bits in the addressing scheme, virtual memory and word length. There are essentially only two common choices of addressing mode: 32-bit or 24-bit. Obviously 32-bit addressing is preferable, as 24-bit addressing limits total (even virtual) memory to 16 megabytes (for byte addressing). Virtual memory allows addressing of more core memory than is physically available, utilizing disk space for the auxiliary memory and swapping things into core as needed. It is of value both for simulating computations on larger machines and simply for doing large computations that do not require all of the data to be in core at once. For certain scientific calculations, having virtual memory is critical, and in this case the transfer rate for the hard disc should be evaluated very carefully. Virtual memory is, in part, also a software issue as well, and so it will be mentioned subsequently with regard to operating systems. Another important hardware issue is the (floating point) word-length. Again there are typically two possibilities, 32-bit and 64-bit. For program development and testing, it is arguable that 32 bits suffice, as long as double-precision is at least available in software. However, this situation may be different for different individuals, and it may change in the near future.

Let us give some simple examples of how the above criteria are used. If one is interested primarily in code *development*, with little need for sophisticated graphical interpretation of data from production runs, then variables one through four are of most importance. On the other hand, someone working with standard software computing on a remote supercomputer might simply need a workstation for interpretation of computational results. In this case, variables four through six are perhaps more important than one or two.The role of variable three for graphics work depends strongly on what is being done; if entire images need to be transferred from memory to the screen, x_3 needs to be very large. In a working situation where one is alone (and say using remote supercomputers via a modem), variable seven may not be critical. But for a group that needs to exchange software, data, etc., frequently between workstations, variable seven ought to be close to unity, if not significantly higher. There is no simple formula for the exact relationship between all of these variables that one can give for all situations. My contention is that each individual (or individual working group) should determine personal weights $w_1, w_2, \ldots, w_7$ regarding the relative value of each of these variables and then solve the linear programming problem:

maximize $\sum_{i=1}^{7} w_i x_i$ subject to the cost of the system X being within budget.

And there may be other factors to be included in the model. It is more important to get a balance of what one needs than to maximize any particular variable. It is certainly feasible to design a workstation around a standard PC for about \$5K with such a balance (but with each $x_i << 1$). It is better to have something that satisfies all of your needs to some degree (albeit slowly) than to give up completely on any important variable.

There are several necessary peripherals not covered by the variable X. Hardcopy devices have dropped in price dramatically recently. Dot-matrix, ink-jet and laser printers offer varying degrees of quality for corresponding prices. Some of these devices allow a limited number of colors. In picking a plain-paper printer, one should consider carefully whether it is compatible with the various software (e.g., for typesetting and graphics – see below) that may want to use it. For true color graphics, it may be advisable to get a separate video printer; models capable of making both 35mm slides and 3 by 4 Polaroids at 1K by 1K resolution are now available for about $5K. Modems are also now quite inexpensive and are essential for communication with the rest of the world, if one is not hard-wired to a high-speed network such as ARPANET (which is of course preferable). Other peripherals that may be useful are "mice," digitizers, "joy sticks," track-balls, voice input/output, and so on.

There is a hidden issue concerning cost of workstations, namely, maintenance contracts. Many of the more powerful workstations require agreements amounting (usually) to about one percent of the list price of the equipment per month, even if it has been bought at a substantial discount. In some cases at academic institutions, this can mean that one pays the same amount for maintenance over a four year period as the original purchase price. This is not the case for more popular computers targeted primarily for the business market (i.e., PC's). We are not arguing that the maintenance costs are not justifiable; indeed, to budget nothing for this item is imprudent even if not required. However, one needs to be aware that the true cost of equipment itself involves two variables, the initial cost and the maintenance cost.

There is another hidden cost related to workstations, namely, the time spent in system management. This can vary from reading the manuals to learn how to format a floppy disc for a PC to complex and time consuming tasks on larger systems that border on the running of a small computer center. The latter certainly is characteristic of a "departmental VAX" and is best done by a dedicated staff person. Even a network of workstations requires a person to be designated "system manager." This person will have to assign sign-on id's and passwords, allocate disc space, back up disc files periodically, install new software, educate new users, and so on. If salary money is not allocated for an appropriate person to fill this role, scientific staff best suited for other tasks will end up having to perform it. This can be a significant waste of human resources.

Software

Software for a workstation is a critical consideration. It is omitted from X because it is hard to quantify, but there are certainly workstations with a lot of software and ones with little. I will try to list important categories of software, without mentioning brand names, and give important considerations where possible. These are as follows:

1. operating system: virtual memory, multi-tasking, multi-user, supports windows?
2. standard ASCII word-processor: full-screen editor, has macros?
3. language compilers: Ada, Algol, APL, Basic, C, Fortran, Forth, Lisp, Pascal, PL/I?
4. communications software: file transfer?
5. graphics packages: standard graphics-terminal emulation?

6. mathematical word-processor or typesetting system: can do all symbols and character sets, has macros?

7. database management

Some explanations of the concepts involved and how the software is used are in order.

A multi-tasking operating system is one that can do more than one thing at once, and a multi-user one allows more than one person to be using the computer at once. Multiple "windows" facilitate keeping track of the various tasks one is doing, by allowing a different "window" for each job on different parts of the screen. A typical situation in code development might be as follows. In debugging or modifying a code to run on a supercomputer, the code is broken into different parts. Imagine the eager programmer who is editing one subroutine while compiling another locally to check for errors. In addition, previously edited and compile-checked routines are being sent over a network to the supercomputer. And data from previous runs could be on the way back. Not to mention a printing job of some sort that is going on as well. Only the compilation is truly computationally intensive; the other tasks involve a great deal of waiting by the cpu and thus can be handled easily without degrading compile time significantly, provided the operating system supports multi-tasking.

The way to write code today is via a word-processing system. It should allow full-screen editing (as opposed to line editing), i.e., to change something just rub out the old and type in the new, with arrow keys governing the movement of the cursor. The ability to copy, move, search and replace are all fairly standard on a wide range of systems. Not so universal are macros, the ability to define some simple symbol to mean a (possibly very long) series of keystrokes. If there is anything that you do repeatedly (and there usually is!), macros can be a big help.

Local language compilers are essential for efficient code debugging even if running programs remotely. It may be useful to have more than one compiler, one that compiles quickly and another that compiles (more slowly) into quick code. Compilers with interactive debuggers are also widely available. Moreover, it may be useful to have compilers for additional languages handy for doing different tasks. For example, graphing some preliminary results might be done most easily in Basic, and writing complicated graphics routines might be best done in C. And a symbol-manipulation program might require a Lisp compiler, or one might choose APL for doing complex algebra. Forth is apparently a very useful language in many contexts, especially for control of laboratory equipment. These are but a few simple examples to indicate how a need for different language compilers might arise.

A communications program should allow you to compute remotely as easily as possible. For example, you should be able to record on disk all that transpired during a session. A critical need is the ability to do "file transfer," which means to transmit a file and to check for errors in transmission as the transfer proceeds (with provision for automatic retransmission if errors are detected). To a large extent, this depends not just on the local workstation, but also on the remote computer system and the telecommunications network that is used. Thus one may need several programs for use with different remote computational systems (one of which could be your local mainframe).

A word-processing system that allows mathematical symbols is not strictly needed for Fortran programming, although it is now possible (e.g., via the WEB system) to write code documentation in natural mathematical symbols. However, preparation of papers for publication is certainly one of a scientist's major tasks, and a workstation should address all needs at once. Similarly, a data-base management program may look out of place, but it can be very useful for mailing lists, bibliographies, etc., at minimal additional cost.

In looking at the list of software, I hope it will become apparent that one should budget a fair amount of money for software. And keep some aside to buy new software as it comes along (this should be thought of as the cost of software maintenance). We have listed seven major categories of software; each of these might represent several packages in itself. Moreover, there will undoubtedly be things of interest to others not listed here. For example, a spreadsheet program is very useful when purchasing further computer systems and can be used in other productive ways as well. Another type of software mentioned by participants at the workshop might be characterized as "productivity tools" or "problem-solving environments." Examples might be things like TKSolver for a PC or MACSYMA on a larger machine (or their many competitors). Software of this type discussed at this workshop include the communications/graphics package of McBryan and the CLAM system described by Gropp.

New Directions

Although we know of no $X = (1,1,...,1)$ workstation for under \$50K, we claim it ought to be possible to construct one, either now or in the near future. Moreover, it is possible to assemble inexpensive systems that have some of these capabilities now, and this represents a major improvement in the price/performance ratio for scientific workstations. Let us consider two examples to illustrate this point. All figures will be approximate, and no vendor names will be given in order to avoid slighting anyone by omission due to my ignorance.

An impressive graphics workstation can be constructed using a lowly PC by adding available single cards containing graphics frame-buffers as follows:

megabytes of display memory	cost
1.0	\$5K
0.5	\$2.4K
0.25	\$2K

These boards come with basic software (GKS primitives, Tektronix emulation, etc.) Corresponding high-resolution monitors can be bought starting around \$1K (for 512 by 512 resolution).

For number-crunching applications, array processor boards are available for the VME bus that have peak computational rates of roughly 15 megaFLOPS (million floating point operations per second) in the price range of \$15K, or one thousand FLOPS per dollar. It is interesting to note that both the Cray 1 and the IBM/PC with an 8087 math co-processor chip yield performance/price ratios of about ten FLOPS per dollar (error bars: a factor of three!). While it is apparently difficult to achieve these peak speeds in general calculations, it should be possible to get multi-megaFLOPS performance relatively cheaply. Moreover, such a computational environment more closely approximates the one that would be used for production runs (e.g., on a Cray or Cyber/ETA machine), and thus would be very useful in the program development and testing stage.

Conclusions

I have listed seven criteria for both hardware and software that are important to consider in designing a workstation for code development and graphical interpretation of data. These variables will affect the overall efficiency of a system, but the true variable of interest is the "throughput" on the problems you wish to solve. Benchmarks are needed for typical tasks, in addition to number-crunching. One might be based on the scenario given above regarding the need for a multi-tasking environment, in which a compilation, file transfer, production run, graphics calculation, printing job and editing are all going on simultaneously. Ideally, you would have the opportunity to try out a system before buying to see how well it fits your needs. One thing to keep in mind, however, is that a workstation changes how one approaches problems, and even the choice of problems. Thus you should choose a workstation that will lead you in a research direction that you want to go, not just get one that appears to automate the tasks you are currently performing.

REFERENCES

[1] Dongarra, J. J. *Performance of various computers using standard linear equation software in a Fortran environment*, Technical Memorandum No. 23, Mathematics and Computer Science Division, Argonne National Laboratory, May 15, 1985.

[2] Fahlman, S. E., and S. P. Harbison, *The SPICE Project*, in D. R. Barstow, H. E. Shrobe and E. Sandewall, eds., **Interactive Programming Environments**, McGraw-Hill, New York, 1984.

[3] Foster, E., *Profs demand cheap super PCs*, **Infoworld**, v. 7, no. 18 (6 May 1985), p. 22.

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 2840	2. GOVT ACCESSION NO. AD-A160 985	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (*and Subtitle*) WHAT KIND OF WORKSTATION SHOULD I BUY? SEVEN-FOLD CRITERIA FOR HARDWARE AND SOFTWARE		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(*s*) L. Ridgway Scott		8. CONTRACT OR GRANT NUMBER(*s*) DAAG29-80-C-0041
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin, 610 Walnut Street, Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit Number 6 - Miscellaneous Topics
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709		12. REPORT DATE July 1985
		13. NUMBER OF PAGES 8
14. MONITORING AGENCY NAME & ADDRESS(*if different from Controlling Office*)		15. SECURITY CLASS. (*of this report*) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

workstation
hardware
software

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

The problem of designing a workstation suitable for an individual scientist or group of scientists will be addressed from an abstract point of view. A set of seven criteria concerning hardware are presented that should be considered when assessing one's needs and comparing various available products. Seven categories of software are also discussed that are critical for someone doing code development, testing and production runs on a remote computer (mainframe or supercomputer). The criteria discussed are intended to be a model checklist relevant especially for people doing code development and production work in

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

20. ABSTRACT - cont'd.

scientific computation. After discussing criteria for evaluating workstations, two examples are given of inexpensive yet powerful systems that should be possible to assemble today.