Trends and Prospects in Computer System Design*†§

Alan Jay Smith
Computer Science Division
EECS Department
University of California
Berkeley, California 94720
USA

Abstract

The directions for computer system development over the next five to ten years are surveyed. The type of system that will come to dominate, or be preferred, over that period will be locally distributed; workstations and intelligent nodes will be connected with local area networks, and will be supported with file servers and larger central computer facilities. Levels of reliability and fault tolerance will increase. Standard software systems, such as Unix‡, will become more prevalent. The performance of high end computer systems will continue to advance, but at a moderate rate. Breakthroughs in multiprocessor performance are unlikely, but multiprocessing (for small numbers of processors) will become more common due to needs for more computation. Specialized hardware will take a more prominent role in certain areas, such as artificial intelligence and graphics.

Throughout the discussion, the hard problems, the research problems and the engineering problems are indicated. Some industrial opportunities are indicated.

---

*The material presented here is based on research supported in part by the National Science Foundation under grant MCS82-02591 and by the Defense Advanced Research Projects Agency under contract N00039-82-C-0235.

†This paper is a summary of an invited presentation at the Korea Institute for Industrial Economics and Technology, June, 1984.

§The opinions expressed in this paper are solely those of the author and do not represent his institution or sponsoring agencies.

‡Unix is a trademark of Bell Laboratories

1. Introduction

Computer systems are now entering their fifth decade; the first electronic computers were developed in the 1940's. Technology has moved from tubes to transistors to integrated circuits and significant progress is now being made each year in increasing the density and performance of VLSI circuits. This improvement in technology is changing the overall design of computer systems, and is expanding their use. In this paper, we outline our view of the directions in which computer systems will develop over the next five to ten years. This time frame is short and we present what we consider to be a realistic view of the changes to be expected.

Much of what will happen in the computer industry over the next few years can be projected directly from current engineering work, from prototypes in laboratories, from the environments in some advanced research laboratories, and from straightforward projections of improvements in certain technologies. These projections indicate what might or might not happen, what is likely to stay in the laboratory, what the hard problems are and what the research questions are. Relevant economic issues are discussed when appropriate.

The development of computer systems is intimately tied to their use; "*solutions looking for problems*" don't usually have an impact until the appropriate problem is found. For that reason, we frame our discussion in the context of various applications. The bulk of our discussion concerns aspects of computer systems for use in scientific and engineering environments. We will talk, within that context, about workstations, local area networks, file servers, massive central computing facilities, software, and fault tolerance. Short discussions of other system aspects such as graphics, artificial intelligence, and data processing follow. A final section comments on possible industrial opportunities.

2. Computer Systems for Scientific and Engineering Environments

Most scientific and engineering environments are now characterized as follows: most scientists and engineers have their own terminals, since terminals are relatively inexpensive. These terminals are connected to small, medium or large computers, depending on the environment. The constraints on the size of the computer relate primarily to cost and available software. The central computer runs the major input/output devices such as printers and tape drives, and manages some number of large disks. The constraints on the other components of the central system are cost and physical facilities (floorspace, power, etc.)

2.1. The Workstation

A major technological advance that is significantly changing the nature of the computer system in a scientific and engineering environment is the fact that it is now economically feasible to give a "*computer*" to a single person, for his/her personal and exclusive use, where that computer can actually do a useful amount of work. In the scientific and engineering environment, such a personal computer is generally called a **workstation** and almost all existing workstations are currently built around *16-bit* microprocessors.

There are several reasons why workstations are desirable. The most important reason is availability; in particular the fact that other users are active on their own workstations does not affect the performance of the local one. Probably the most serious problem of a shared computer is the congestion caused by heavy use during the day; this is avoided on the workstation. (It is well known that a shared computer "runs faster" at night due to the lack of congestion, and "very slowly" during the day. On the other hand, *a slow personal computer runs just as slowly at night as during the day.*) The workstation also satisfies a

human need for local control, and may provide superior privacy for stored data. If the work site is distant from the central machine, the use of the work station may cut reliance on a low bandwidth connection such as a telephone line.

2.1.1. Workstation Processors and Computer Power

Workstations now available are mostly based on 16-bit microprocessors. The SUN workstation uses the Motorola 68000, as do a number of other machine either available or under development, including a machine by Metheus and one by Apollo. The PERQ uses AMD 2900 series bit slice products. Digital Equipment Corporation has workstations based on the VAX architecture. Less powerful personal computers, sometimes used as workstations, include the IBM PC, which is based on the Intel 8086 family of microprocessors. The 68000 provides about .25 MIP (millions of instructions per second), as compared to about .75 MIP for the DEC VAX 11/780, 2.7 MIPs for the IBM 370/168, 4.0 MIPs for the Amdahl 470V/6, and 12-14 MIPs for the Amdahl 580 and IBM 3081. (The instruction sets of these various machines differ, but the figures presented have been roughly normalized to the IBM 370 instruction set, and should be comparable.)

In addition to the basic computational facility, many workstations are available with hard or floppy disks, high resolution (800 x1000 pixels) screens, and pointing devices such as mice or light pens. Some, such as the workstation from Metheus, have high resolution color displays, hard disks and tape drives.

For many of the tasks normally done on a computer, .25 MIP is quite adequate. Editing and data entry are trivial. Small computations and compilations can easily be accomplished locally. Compiling a 500 line Fortran program took the author about 1.5 minutes on a SUN workstation. Serious computation, however, proceeds too slowly; a Fortran program which ran in just over a minute on one of the IBM 3081 dual processors ran just over an hour on a SUN. (The software on the IBM machine was much more efficient, accounting for an additional ratio of two over the performance figures above.)

The level of performance available from a workstation will increase with the speed of high performance microprocessors. In the 1985-86 time frame, work station performance should increase to over .5 MIP with the availability of the 68020, the Zilog Z80000, the DEC micro-vax(es), and the National 32132. (Many of these parts are specified with various clock rates, and the performance level will depend on when and if those clock rates are achieved.) By 1987-89, individual workstations based on microprocessors will be available that run at 3-5 MIPs, and workstations will be available in that time frame that run at close to 10 MIPS, using LSI and MSI higher speed technologies. For comparison, the largest "commercial" mainframe in 1973 was the IBM 370/168, which with a primarily commercial workload runs at about 2.7 MIPs. The IBM 360/91, one of the largest scientific mainframes of the 1960s (along with the CDC 6600 and 7600) performed comparably to the 370/168. Workstations in this performance range will permit serious scientific computation.

2.1.2. Workstation Disks and Displays

A workstation is of only limited use without significant local storage. Small hard disks, with diameters of 5.5 to 8 inches and holding 50-200 megabytes, are already available, and it is even possible to buy a 450 megabyte hard disk that takes less than 4 cubic feet (.11 cubic meter) of rack space, for about $10,000. Improvements in disk technology are proceeding at a regular rate, and density increases of a factor of 2 to 4 should be available by the end of the decade. (Some of these disks are noisy, however, and cannot be placed in an office.)

The better workstations currently are available with high resolution displays. Such displays are necessary for graphics, for windows, and for the display of various character fonts. High resolution monitors, many with color, will become more available. If and when high resolution commercial television becomes available, high resolution monitors will become the standard, due to the low prices available from economies of scale.

Workstation software is, of course, a major issue, but will be discussed below in our general discussion of software.

2.1.3. Workstation Problems

Workstations are becoming more generally available, but there are some factors which are limiting or slowing their availability. First, they are expensive. A typical high performance workstation costs about $15,000 - $20,000 without a disk, and more with a disk. Second, the methods to integrate an environment consisting of workstations and central computation facilities are not yet fully satisfactory. Third, reliability can be a problem; centralized computer systems take steps to ensure that files are not lost (backups, etc.), but these steps are rather troublesome without a paid staff of operators. Finally, a workstation with a disk is large and noisy, and cabling can be difficult and cumbersome to install, so that a private office is not a pleasant location for a workstation. All four of these problems will be overcome in time, and in ten years, we predict that workstations will be as common as terminals are now.

2.2. Local Area Networks

As explained above, changes in technology are making it cost effective to have numerous small but powerful computing nodes, such as workstations and distributed minicomputers. Effective use of these distributed processing nodes requires effective communication among them. The most important use for communication is to permit the sharing of files, data bases and I/O devices; secondary uses include mail, paging, backup, load sharing, and the sharing of large computers. Such communication is normally accomplished with a *local area network (LAN)* such as **Ethernet**.

Several local area nets are available. Ethernet uses a contention protocol and is supported by DEC and Xerox, among others. Token passing nets are straightforward. Broadband networks such as Wangnet are also available. Bandwidth on Ethernet is 10 megabits/second, and some other nets have even higher bandwidth; thus far, bandwidth has not been a limiting factor.

2.2.1. Local Area Network Problems

Although there is nothing *hard* about local networks (LANs), there are several reasons why they haven't yet become widespread. The most important reason is **standardization and compatibility**. Local area networks often have to or should tie together pieces of equipment from different manufacturers, and in many cases, the systems are incompatible. DECNET is not the same as WANGNET which isn't the same as SNA. (These aren't just LANs, however, but are more general communication networks.) IBM recently indicated that its local area network would not be available for 2-3 years, although it announced the physical cabling, which disappointed many people hoping that the IBM product would become a standard. Once a standard exists, various (smaller) vendors can develop products to interface to it; only a small number of major manufacturers can afford to develop both a network and a variety of products which use it.

A second problem with local networks is the software overhead of using them. Most operating system software is layered, and the use of the local

network often involves the execution of thousands of instructions. (There are one or two exceptions to this, such as the Locus system at UCLA.) This high overhead means that file transfers are slow, and paging is very inefficient if done over the net. This problem will only be overcome when the system software is rewritten to provide very efficient local net access.

A third problem is cost; the interface to the local net may involve a non-trivial amount of hardware and may cost hundreds or even thousands of dollars. That cost is significant if the local station is a small personal computer such as an IBM PC or an Apple II; it is less important if a powerful and expensive workstation is used. The cost problem will become much less significant with the creation and volume manufacturing of VLSI chips which implement the interface to the net.

Bandwidth is not currently a problem on local area networks. The amount of material to be transferred is small and software overheads are often so high that it is impossible to saturate the network. Bandwidth may become more of a problem as the power of the local nodes increases, the number of nodes increases, and software becomes more efficient. Technical solutions exist, however, to extend the bandwidth sufficiently to overcome any problems likely to occur in the next 10 or more years. Broadband networks have enormous bandwidth, and fiber optic cables can be used in place of metalic conductors.

2.2.2. Local Nets - Predictions

We believe that local area networks will be widely installed in the next five years; the major impediment is compatibility and standardization. We also predict that whatever standard becomes initially established will be found to be inadequate over the subsequent 5 to 15 years, and a new standard architecture will then appear.

2.3. The Central Computation Facility

A locally distributed computer system needs a central computation facility (CCF) for several reasons. Many or all of the nodes are likely to lack a full set of input/output devices, such as printers and tape drives; files can be spooled from the nodes to the central site for printing, for example. The central site will likely have a large shared file system, far superior to what would be available on any one node. Finally, the central node could contain more and more specialized processing power - e.g. a large mainframe and/or a database machine. In this section, we discuss two aspects of the CCF- the file server and high performance computing.

2.3.1. File Servers

A file server is a machine or set of machines with some amount of file storage, usually several large disks, which manages a file system shared in a distributed environment. File service may be one of several functions performed by the CCF, or there may be one or more dedicated machines that act as the file server.

There are four reasons to use a file server: (1) There are many files that are shared among many or most nodes of the distributed system, including libraries, compilers and databases. The file server (FS) can coordinate access, to provide consistency, and can control the proliferation of identical copies, in order to minimize the waste of storage. (2) The FS can manage a large number of disks, which can be used and allocated more efficiently by the central FS than they would be if the disks were distributed among the nodes. (3) The FS can be associated with resources that could not be economically provided at each

individual node. In particular, steps can be taken at the file server, such as periodic dumps by a paid operator, or file duplication across disk drives, that wouldn't be seriously considered at the nodes. (4) The file server may, including its disks and other devices, require an environment that is impractical at the local nodes. This environment may include cooling, sound insulation, power and large amounts of floor space.

There are also two possible problems with a file server. First, the use of a file server is potentially inefficient, since requests to read and write are sent over a network, with overhead at both ends. Second, there can be a consistency problem - files or parts of files can be kept in one or more nodes, and/or at the file server, and at each of these places there may be a disk cache. Even though the file server may manage the locks correctly (one writer or many readers), it needs to keep track of parts of files kept elsewhere in the system.

2.3.2. High Performance Computing

In this section, we first discuss the need for high performance computing by outlining some of the applications. Then we discuss the techniques that are being used or have been proposed to achieve high and higher performance.

2.3.2.1. Applications

The applications for high performance computing seem to fall into three categories. The first and simplest category is that which requires large aggregate processing power, but does not necessarily need a fast processor. A bank would fall into this category; it has to process a very large number of checks overnight, but whether all the checks are done on one machine, or whether 10 machines are used is largely immaterial. (The database is shared between machines, however, which can lead to inefficiencies.)

The second area of application of high performance computing is numerical simulation and related numerical problems. This might include fluid flow, as in designing airplane wings or ships, and weather prediction. Analyzing seismic data from oil prospecting teams would be another use. Graphics and image processing to some extent fall into the same category of numerical problems. Sometimes parts of these problems can be solved in parallel, but parts of the computation are sequential; sometimes all of the computation is sequential.

Artificial Intelligence is the third area of application for high performance computing, and is also a (or the) primary target of Japan's fifth generation computer project. AI applications are primarily characterized by decision oriented code with little in the way of numerical computation or looping. We comment further on this topic in a later section.

2.3.2.2. Techniques for Improved Performance in Existing Computer Architectures

There are hundreds of billions of dollars of software that runs on existing computer architectures, such as IBM 370s (including 360s, 3081s, etc.), DEC VAXes, etc. Users of that software would like to continue using it, but with faster and more powerful hardware; thus there is the need to make faster implementations of existing machine architectures.

The general list of techniques available to speed up a given machine architecture are well known. Cache memories can be made larger and more efficient. The degree of pipelining (sequential overlapping of instructions) can be increased. Inefficiencies in the operation of specific machines can be determined by measurement, and can then be eliminated one by one. These

techniques, however, can only be exploited economically to a certain extent.

The other method for speeding up a given machine architecture is to improve the speed of the logic circuits. This is by far the most promising approach and will yield most of the improvement to be obtained over the next few years. Improvements in semiconductor technology will increase the number of circuits per chip and will decrease the switching time per logic gate.

The overall rate of improvement in CPU performance for a given architecture has become rather gradual. The IBM 370/168 was first available in 1973 and the 3081 in about 1982. Their performance ratio, per processor, is about 2.6, which indicates an annual improvement of about 11%. (The 3081 is a dual processor machine.) It would be unrealistic to expect more than 10% to 20% improvement per year over the next decade.

2.3.2.3. Architectural Changes to Increase CPU Performance

2.3.2.3.1. New Instruction Sets

In the 1970's, the prevailing wisdom in designing computer instruction sets was that complicated operations should be made into single instructions; this was believed to have two advantages: those operations would execute quickly, and some complexity would be removed from the software. More recently, the general opinion has changed drastically; further thought and observation has shown that a side effect of implementing complex operations in one (or a few) instructions, is to slow down the operation of simple instructions. It has been further observed that the simple operations occur very much more frequently than these complex ones. Thus it is now believed that much higher performance can be obtained with so-called **RISC** or **reduced instruction set computers**, in which only the most basic instructions are implemented directly in the hardware. Those instructions would include load, store, add, and so on, but *not* polynomial evaluate, add to storage, queue, etc. It should be possible to get a factor of 2 to 4 over existing commercial architectures (e.g. the IBM 370) by careful instruction set design.

2.3.2.3.2. Parallelism

As noted above, there are limited possibilities for improved CPU performance through faster sequential instruction execution. Improvements beyond that point must be obtained with parallelism. Several types of parallelism are possible and we discuss each briefly.

2.3.2.3.3. Vectorization

It happens that for many numerical algorithms, the sequence of operations can be reorganized so that many can be represented as element by element operations on vectors, such as addition, subtraction, multiplication (dot product) and division. These vector operations can be performed very efficiently in properly designed machines, such as the Cray-I. Vector operations and instructions can be added to existing architectures. Vector architectures, of course, are only useful for problems which can be extensively vectorized; even if 90% of the problem can be vectorized, the other 10% can and may well be the bottleneck.

2.3.2.3.4. SIMD Architectures

SIMD means *Single Instruction Stream, Multiple Data Stream*; an example of a SIMD architecture was the Illiac. SIMD architectures operate in lock-step, with a central control processor controlling a set of slave processors, each of

which does the same operation to a different data item. SIMD architectures are able to operate efficiently with only a small class of problems, and are not likely to be applicable to a wide or general class of problems.

2.3.2.3.5. Data Flow Architectures

Data flow architectures are currently primarily a theoretical concept in which the processor consists of a large number of simple processing elements. Each processing element is given an operation, and input and output "cells" are identified. When the inputs are ready (written into), then the element operates and puts its output in the output cell. The potential is for very high overall throughput rates, if enough operations are ready and can be performed at once. At this time, no significant data flow machine has actually been built, although a small experimental one is reportedly running at the University of Manchester. Data flow machines have been proposed, discussed and studied for many (~15) years with little progress; we consider the approach of fine grained dataflow architecture to be at best a laboratory curiousity for at least the next decade.

2.3.2.3.6. Multiprocessor and Multicomputer Architectures

If there are N computers and each operates at M MIPs, it is a logical conclusion to note that in the aggregate, they (can) operate at N*M MIPs. The idea is to take a problem, decompose it so that at any given time there are at least N pieces which can operate in parallel on each of the N processors, and then assign the pieces to the processors dynamically as each is ready to execute. There are many difficulties with this approach. (1) At some or many times, there may be considerably less than N pieces which can be done in parallel; if so, the overall speed of the computation becomes limited by the sequential or almost sequential sections. (2) While different computations can proceed in parallel, they may share inputs, and conflicts in accessing those inputs may lead to significant inefficiencies. For example, if several processors are reading from the same memory, there will be memory interference. (3) There are likely to be significant overheads in breaking off pieces of the computation, assigning them to processors, and collecting the results. This last item suggests that the pieces be relatively large - e.g. at least thousands of instructions, not 5, 10 or 100.

2.3.2.4. Preferred Direction for General Purpose High Performance Computing

We believe that multicomputer (no shared memory) or multiprocessor (shared memory) architectures are likely to be the preferred mechanism for high performance over the next decade. Further, due to the three problems noted immediately above, we believe that a small number of powerful processors will be *much* more effective than a large number of slow processors. (An analogy is the committee: *a large committee of stupid people is not an effective substitute for a small number of smart people.*)

There is another advantage to a general purpose multiprocessor or multicomputer system. Such a system can either process a workload consisting of many normal sequential jobs, or can process a workload of jobs which fragment themselves into smaller parts, each of which can be assigned to its own processor.

2.3.2.5. Algorithm Development

It is important to note the close coupling between computer architecture and algorithms. Many of the high performance architectures described above are specialized and algorithms must be designed carefully to take advantage of them. The development of such algorithms is as important as the design of

machines to run the code.

2.4. Fault Tolerance

Fault tolerance is an obviously desirable attribute of a computer system. Computer failures can cause problems ranging from inconvenience to disaster, and are usually extremely annoying. There are two ways to obtain a reasonably reliable computer system. The *"right"* way is to design the system from the start to be fault tolerant, with redundant hardware and software. Such systems are currently available from both Tandem (since 1976) and Stratus (since 1983), and are under development by a number of small companies. (ATT Information Systems has recently announced some reliable machines.) The problems with these systems are that (a) they are not compatible with existing systems and architectures, so that they cannot substitute directly for existing machines, and (b) they are expensive due to the redundancy.

The second way to obtain reliability is to gradually improve an existing architecture. This can be done by recoding the software to provide redundance and fault checking, and putting in some redundancy and fault checking in the hardware. This approach is unlikely to ever be completely successful, but for non-critical applications, it can be good enough.

As might be expected, all major manufacturers of computer systems are working hard to improve the reliability of their products. Software is going through more extensive testing and contains self checking and recovery code. Hardware reliability is increasing both due to more careful manufacturing, and due to decreasing chip counts; chip counts decrease as more circuits can be placed on each chip. It is the general perception of this author, unsupported by any measurements, that computer system reliability has been generally increasing, and is considerably better than it used to be.

We believe that the following will occur: (a) Reliability of existing nonredundant systems will continue to increase, although not nearly to the level attained by genuinely reliable systems. (b) One or more major vendors will introduce a reliable computer system. That system will be largely but not fully compatible with its existing product line.

2.5. Software

Software combines with hardware to make a computer system, and software is no less important than hardware. In this section, we discuss some of the forthcoming developments or non-developments in software.

2.5.1. Software Changes and Evolution

The most important thing to remember about software is its **stability**. Everything that the user wants to do is accomplished through software - the instructions that perform the necessary function. Everything that the user interfaces to is controlled by software - including the editor on which this paper is being written, the compiler which translates programs, and the operating system which controls program execution. The user therefore becomes used to a particular interface, and his programs will often run only with a specific operating system. If the software changes significantly, the user will have to rewrite his applications programs, and will have to learn a new interface. Users are generally unwilling (!) to do either, and thus there is a very strong constraint on any changes to existing system software, such as compilers, utilities, and the operating system. Such changes as are made need to be transparent to the user to the greatest extent possible. Thus existing systems will change only in an evolutionary way, and any software system with a large enough user base can be

expected to remain in use for a very long time.

There are a few changes to existing software systems which can be expected to occur, but most of these are in the nature of extensions, not significantly incompatible modifications.

The size of the address space is a crucial aspect of any useful computer system. Extending the address space was the reason for the creation of the VAX line of computers by Digital Equipment, and was the major reason for the changes in the architecture in going from the IBM 370 to the architecture of the 308x machines. In the latter case, it was necessary to rewrite the major operating system MVS, now known as MVS/XA, to be able to take advantage of the 32 bits of addressing as compared to the 24 bits available previously. We can expect that any existing major computer system with less than 32 bits of addressing and without virtual memory will be extended to include both features; the operating system will have to be modified accordingly. If the operating system limits the number and variety of input/output devices, that restriction will also have to be lessened through modifications.

As noted above, the operating system can be expected to be modified to provide increased fault tolerance and reliability. In the case of failure, it will become increasingly easy to restart the machine and have it resume where it left off, with minimal loss of work in progress, and likely no loss of files or portions of the database.

The system software will also be modified to accommodate locally distributed systems, and multicomputer and multiprocessor systems such as have been described throughout much of this paper. This modification will have several aspects: (1) The file system must be extended so as to provide a way to reference files that are resident on either another machine or at the file server. The user may or may not have to be aware of which machine holds the file, and in general, should not have to be aware of that. (2) The operating system must be modified to permit processes to be assigned to any of several processors for execution, rather than just permitting execution on the local processor. (3) The software must be modified to permit the *efficient* creation and use of small processes which can be used in parallel to perform subtasks.

2.5.2. Unix*

The fact that users cannot easily migrate from one set of hardware and software to a different and incompatible set means that any new system design will be able to acquire a customer base only slowly, as the new customers build *new* applications to run on the new system. Further, the new system will acquire few if any new customers unless a reasonably good set of software is provided, including a sophisticated operating system, a reasonable set of debugged and optimized compilers, an editor, an assembler, a text editor, etc. It costs millions of dollars to develop even a minimal set of software, and it takes several years and tens to hundreds of millions of dollars to develop a satisfactory set of software. (The cost to develop the system software available from IBM has been tens of billions of dollars.) The implication of this line of reasoning is that only the largest companies can afford to consider developing a completely new hardware/software architecture, unless there are special considerations. One such special consideration is fault tolerance, as noted above.

There is one good alternative to developing a completely new hardware/software architecture. The Unix operating system, developed and marketed by

*Unix is a trademark of Bell Laboratories

Bell Laboratories, is relatively *portable*, and can be converted (ported) from one computer system to another in *only* a few man-years, as compared to the hundreds of man-years to develop a minimal set of software, and the thousands to develop a comparable set to that provided by Unix or any major system. Unix is written in C, a high level language, and thus to port Unix from one system to another, there are two primary steps: (1) The C compiler must be converted to generate code for the new machine. The compiler itself is written in C, so once the code generation part of the compiler is rewritten, the new version of the compiler is compiled on the old machine under the old compiler. The new version of the compiler is then run on the old machine, and it is used to compile itself. The output is a machine language version of a compiler for C to run on the new machine. (2) The machine dependent parts of Unix must be rewritten to conform to the new architecture. Only a small fraction of the operating system is machine dependent.

Associated with Unix, and available at the same time, is an enormous set of utility programs, including compilers, assemblers, text editors, text processing programs, type setting programs, etc. Thus for a relatively small effort - porting Unix to a new machine - a full set of mature and sophisticated software is immediately available. The availability of Unix thus makes it feasible to create a new machine architecture and market it without having to plan on creating a full set of system software. Just this approach has been taken by many of the workstation vendors, such as SUN Microsystems.

Unix has a number of advantages. It is very versatile and a very large variety of useful utilities are available. In some ways, it is very consistent; a line of reasoning of the following type is usually successful: <wouldn't it be nice if I could do X>; <I bet that I can accomplish X by doing P, Q and R, which have that effect in a different context>. In comparison to major vendor operating systems such as MVS (IBM) or VMS (DEC), it tends to be simple to do simple things. (It is *not* simple relative to much smaller systems.) Unix is a relatively *open* system; it is easy to add things to Unix. One reason why Unix is so popular among sophisticated users is that for over 10 years other sophisticated users have been using Unix and adding whatever they thought would be useful and nice to have.

Unix also has *disadvantages*. The system is hard to learn, and its scope and complexity mean that even after years of use, one is regularly discovering new features and is regularly learning how to do new things. Its user interface is rather cryptic, and the naming of commands is obscure. Some of the commands can do great damage, and they execute without issuing warnings. The Unix code is not especially efficient, and for example, Fortran programs run much (e.g. 30%) more slowly under Unix (with the Unix f77 compiler) than under DEC's VMS on the VAX.

Unix has recently been adopted by DEC as Ultrix, and IBM has made available two versions of Unix on its small machines. With the support of such major vendors, we predict that the disadvantages of Unix noted above will diminish. The system will be recoded for greater efficiency. New, safer, and more obvious commands will be created as alternatives to the cryptic and dangerous ones that now exist and will largely come to supplant them. The system will be extended to include a number of features including multiple windows and mice, and a distributed version with a transparent file system, process migration, and fault recovery will be built. Over the next few years, Unix will achieve greater and greater penetration into the market and will become a quasi standard for sophisticated software development environments.

It is also worth noting that like all software, Unix will eventually be surpassed. The difficulty is to guess from where the replacement will come. The

cost of developing a major operating system and associated system software is so large (over $100,000,000), that it is hard to imagine a vendor willingly undertaking such an exercise without compelling reasons; it is also worth noting that major vendors have been relatively unsuccessful in producing high quality software by the "direct" route of defining such a product and then building it. Many or most of the more desirable systems have developed from a small, modest base, built by a small number of very talented individuals for their own use, which is just the way Unix was built. IBM's VM system similarly came from the CP/CMS system at the Cambridge Scientific Center. (Please remember that the initial base and the final result differ *greatly* in scope, polish and reliability.) We believe that over the next 10-15 years, a contender to Unix will appear by the same route.

3. Other System Types

3.1. Data Processing and Data Base Systems

General data processing, including payroll, assorted engineering calculations, inventory, financial analyses, and so on, are often found running together with data base applications on central corporate mainframes; we therefore consider the two together and refer to them as DP/DBS systems.

Both data processing and data base systems tend to have a lot of small and medium size jobs or transactions, and most of those jobs would run more than fast enough on an otherwise unused single machine; this is in contrast to those applications requiring high performance computing, where on the fastest machines, jobs can run for days. The need in the DP/DBS environment is for aggregate processing power, and that can be, and usually is obtained by using a number of machines.

The problems associated with DP/DBS environments include those mentioned earlier in discussing multicomputer systems. Efficient and consistent sharing of the file and database system is often difficult. Mundane problems of floor space (for processors and disk drives), power and cooling frequently cause users a great deal of distress. Backing up or otherwise making reliable a database that can be 50 or 100 gigabytes (or more) can be difficult and cumbersome.

Some DP/DBS environments can benefit from distributed environments. For example, a large bank with many branches would function best with a distributed database, with much of the account information being kept both locally and at the central site. We understand that at least one major bank operates in just this way, and to minimize overhead, reconciles the remote information and the central site information overnight, rather than immediately after a transaction. Another situation that works well in a distributed manner is a very large inventory system, with subareas or local warehouses.

Over the last 10 or more years, there have been persistent proposals for "database machines"; i.e. machines whose architecture and software is specifically optimized for database use. There have even been a few products, such as Britton Lee's database machine. In general, the current market for such database machines is small. There are several reasons why database machines have not become popular. Most important, it isn't clear that they fill a particular need. There exist many data structure techniques to produce efficient access to large databases, and in the context of the overall system, database machines don't appear to produce significant performance gains. These data structure techniques are often more efficient than brute force techniques, given the large access times to secondary storage. Further, the database machines are specialized, as opposed to the general purpose CPU, and thus

are not well suited, if available at all, to do general purpose data processing. Also, they are different, requiring additional expertise, parts, etc. Finally, no such machine is available from a major vendor; thus the idea lacks credibility.

We don't see any particular *need* for a database machine, and don't expect that such a product would be especially successful. If a major vendor should build one, such as IBM, it would likely do well enough, but perhaps not well enough to justify diversion of scarce engineering talent from other projects. (There is at least one small company currently designing a workstation explicitly designed for accessing databases. It isn't clear to what extent the special features of this product are implemented in hardware, nor whether those hardware features are worthwhile. It is also to be determined as to whether there exists a market for this product.)

3.2. Graphics

There are a number of applications for graphics. One of the simpler ones includes Computer Aided Design (CAD), in which graphical representations of objects need to be specified, displayed, rotated, and changed. At the complex or difficult end of the spectrum is the generation of movies via graphics. The former is simple because only one or a few images are needed, and they can be schematic rather than realistic. The latter is very difficult because many images are needed (e.g. 16 per second of film), each must be realistic, and very high resolution (e.g. 2000 x 3000 pixels per frame or better) is required. Other areas of graphics use are simulators (e.g. flight simulators) and video games.

Graphics seems to benefit from specialized hardware. In addition to high resolution color displays, items such as frame buffers and character generator chips seem to be useful and cost effective. Also, many graphics applications seem to require massive amounts of computer power. Consider, for example, making a movie entirely by computer. Assume that there are 2000 x 3000 pixels per frame, there are 16 frames per second, there are 120 minutes of film, and there are 100 instructions executed per pixel. Then such a film requires about 2000 hours on a 10MIP processor. (The 100 instructions per pixel is a very modest figure.)

We believe that much specialized hardware will be created to support graphics. The simple end of the graphics market will expand significantly, since equipment costs can be kept moderate and affordable. The sophisticated end of the market will expand more slowly, due to the small number of applications that can justify the high costs.

3.3. Artificial Intelligence

We use the term Artificial Intelligence (AI) to refer to applications in which people are generally or exclusively used, rather than computers. (It can also be said that AI includes applications that we cannot yet solve.) We include here vision, robotics, expert systems, theorem proving, speech recognition, language understanding, etc. The state of the art of AI is such that many of these functions can be done in a very rudimentary way, if at all, by computers. The algorithms to perform these functions require massive amounts of computer power, and CPU time is one of the factors limiting the obtainable results; it isn't clear, however, how much better one would do with virtually unlimited CPU time. These problems are quite *hard*, and they are not very well understood. These are things that people do well, and machines do poorly, if at all.

The domain of AI problems is the primary target of Japan's Fifth Generation Computer project (FGCP). Their idea is to build machines that do "logical inferences" very quickly, several orders of magnitude faster than current machines,

and with significant parallelism. This approach is in line with the opinion of some or all of the AI community; some AI researchers use, design and/or build machines which are specialized to execute "AI programs" quickly.

There are two problems here with the FGCP. First, the domain of application is not well understood, and it isn't clear how much more will be accomplished with faster machines. Second, it isn't clear how much parallelism can actually be obtained in solving problems in this domain. (The human brain seems to solve these problems in a very parallel way, using special purpose hardware; we don't, however, understand how the brain works.)

We believe that AI applications will become increasingly important, and that software and hardware will develop rapidly. Progress, however, will likely be slow and gradual, and we don't expect any quick breakthroughs. The most important areas of application will be expert systems, robotics and vision, where there are clear and pressing needs and substantial demand. The expert system area, in particular, is sufficiently well understood that useful products can be produced in a straightforward way.

4. Industrial Opportunities

There are a number of opportunities for industrial ventures in the computer system area. We discuss these below.

(1) There are many components of computer systems where manufacturing cost, as determined by efficient engineering, efficient manufacturing, low labor costs and economies of scale, is already the dominant factor. Examples in this category are floppy disk drives, small hard disk drives, personal computers, keyboards, video games and video monitors. These items are already being made in either very highly automated factories (as by Apple) or in areas of low labor cost.

(2) There are products where we believe that there is very substantial price elasticity. The most obvious such product is a work station. Currently, a work station costs in the neighborhood of $10,000- $25,000, which is a significant amount of money. If a comparable work station were available for $5000, we believe that the market would expand greatly, and if the cost were near $2000, millions of units could be sold.

(3) The successful creation of software requires a close connection to and a clear understanding of the application. For that reason, almost all software in use in the United States was written in the USA. Similarly, there must be many many applications throughout the world for which software can only be effectively written by those familiar with the problems.

(4) There should also be a market for the adaptation of software. The creation of software is time consuming, labor intensive and expensive, and it is often much faster and cheaper to adapt or modify a similar product than to build a new one. For example, a database system could be modified to operate with commands in a different language and with a specialized keyboard. Modification of software under license from the original developer should be a profitable project.

5. Conclusions

There are a number of ways in which computer systems are currently advancing. Personal computers and workstations are becoming increasingly popular and powerful. At the other end of the power spectrum, resources and energy are being devoted to designing and implementing high performance computers. Local area networks will come into use to unite the various aspects of

the computer system. Operating system software is evolving to support distributed systems, with increased functionality, increased reliability and improved user interfaces. The availability of Unix is permitting the development of new architectures without the expense of a major software effort.

Computer applications will also continue to increase. Consumer and industrial products will contain more and faster microprocessors. Applications now not well understood, such as those referred to by the term "artificial intelligence", will slowly advance. People will come to more and more depend on computers to serve them and perform essential functions.