$$\binom{M}{p} \geq 14{,}000 \text{ and } 28p < \tfrac{1}{2}M,$$

since the number of keys in the DR&D system is close to 14,000, and the average number of keys per compound (record) is 28. The minimum value of M that will satisfy these two inequations is $M = 167$, where $p = 2$, which yields a key vocabulary of 14,000. However, the mask size of 167 turns out to be so close to the minimum value of M for $p = 3$, which is 168, that the optimal design values would actually be selected as

$$M = 168$$

$$p = 3$$

$$\binom{M}{p} = 2.46 \times 10^6$$

To get an idea of search file size in order to perform the key logic (exclusive of the connection tables or notations required to perform iterative search), assume that a record in the sequential file required 4 bytes for a compound identifier and 21 bytes for the 168 bit mask, yielding a fixed length record of 25 bytes. The total file size for 250,000 compounds would be 6.25 megabytes. A conservative estimate for the time to process the 168 bit mask would be 300 microseconds, which would yield a file search time of 75 CPU seconds. The time to transfer the search file from disk to core would be around 15 seconds, if head movement could be avoided; hence, the elapsed time would be at least 75 seconds, and in a normal multiprogramming mix might run 10 to 20 times as much. To this must be added the iterative (atom by atom) search time, and, since superimposed codes add their own coded false drops to those injected by the chemical screens, the amount of iterative searching would increase.

In conclusion, the superimposed code, employed in a sequential, highly compacted search file bears some further investigation even though at this moment it would appear as though, for interactive search of large files, it is not yet as efficient as the Hybrid Inverted list.

A particularly severe problem that the interactive system must face in the substructure search application is the "don't care." This condition is manifest as a series of OR's (disjunctions). A few "don't cares" each with several possible values can generate several hundred combinations, yielding an extensive logical sum of terms. The elapsed processing time in such a situation can be very long, as indicated above, particularly in a multiprogrammed operating environment. Such a system must be further optimized by a logic processor that will operate either upon list pointers or bit maps depending upon list length, and by retaining a bit map in core during the entire evaluation of a long sum of terms. This would be equivalent to retention of the top of the postfix logic processing stack in core rather than shuttling it back and forth to disk with each logic operation. This kind of optimization presents a time (CPU and channel) *vs.* space (core) trade-off. A file of a quarter million records must dedicate 32,000 bytes of core for the top of stack, while a file of 2.5 million must dedicate 320,000 bytes. Clearly, at some point some combination of zero compression, disk shuttling, and pointer list (as opposed to bit map) processing must be employed. These are questions and techniques for the future of such system developments to explore.

LITERATURE CITED

(1) Prywes, N. S., and Gray, H. J., *et al.*, "The Multi-List Type Associative Memory," Proceedings of Symposium on Gigacycle Computing Systems, AIEEE Publication, No. S-136, Jan 1962, pp 87–107.

(2) Lefkovitz, D., "File Structures for On-Line Systems," Spartan Books, New York, N. Y., 1969.

Large Data Base at the Lawrence Livermore Laboratory†

JOHN G. FLETCHER

Lawrence Livermore Laboratory, University of California, Livermore, California

Received December 2, 1974

The Lawrence Livermore Laboratory Octopus network includes a central storage facility of over 10^{12} bits. This facility is accessed through a directory mechanism which permits flexible organization and general sharing of files by many time-sharing users. The chief problems of the system are not unusual: the maintenance of high transfer rates, of reliability and availability, and of sufficient storage capacity.

The Lawrence Livermore Laboratory of the University of California (LLL) has one of the largest (if not the largest) concentrations of computing capability in the world. Processors include four CDC 7600's, one CDC 6600, two DEC PDP-10's, and numerous smaller machines; two CDC STAR-100's are on order. Input–output devices include two 15,000 lines/min Honeywell printers, two III FR-80 microfilm recorders, over 100 alphanumeric and raster display monitors, and over 600 teletypewriters and other interactive terminals. Data storage facilities include a terabit (10^{12} bit) IBM photodigital store, an IBM Data Cell, about 20 CDC 844 disk packs, and several other disks with transfer rates up to 40 MHz: there is also a vault containing over 30,000 magnetic tapes.

All these facilities (except the tape vault) are joined together into a single interconnected computer network called Octopus.[1] The organization of Octopus can be approximately characterized as a superposition of about eight *subnetworks*. Each subnetwork is centered on a computer called the *concentrator* of the subnetwork and performs a

† Presented in the "Conference on Large Data Bases," sponsored by the NAS/NRC Committee of Chemical Information, National Academy of Sciences, May 22–23, 1974. This work performed under the auspices of the U.S. Atomic Commission.

single function in service of the large (CDC) *worker* computers which execute users' programs in a multiprogrammed, interactive, time-shared fashion. Each subnetwork concentrator is connected to all the worker computers and to whatever I/O devices or storage media are appropriate to the function of the subnetwork. Subnetworks currently implemented or being implemented include one for collecting experimental data in real time, one for I/O at remote noninteractive stations, one for high-speed printing and microfilm recording, one for softcopy display output, two for interactive I/O, and two for controlling central data storage. The advantage of the subnetwork idea is that the network does not have all its eggs in one basket. If there is trouble with one concentrator, then the other subnetworks continue to operate. At most one service is lost, not all services.

The advantages of having a network, rather than computers separately standing alone, are, first, that the terminals are universally usable. One can sit at any terminal and access any of the large machines. Second, certain unique equipment, like 15,000-lines/min printers, can be accessed from any computer. While the Laboratory can afford two 15,000-lines/min printers, it might not be able to afford five, so as to provide one for each worker computer. Third, there is a single data base directly accessible from all the computers. Fourth, there is the possibility of cooperation among the computers; that is, if one has a single problem that is too big for one 7600, one could divide it between two of them in some fashion.

Octopus is organized as a general purpose computer utility. A user (LLL scientific, clerical, or administrative employee) can invoke the execution of a program on any worker computer from any interactive terminal. The program can in turn utilize any of the network's facilities. A user's programs can be written in any language of the user's choosing and can perform extremely varied tasks, including numerical computation, language processing, text editing, and information retrieval. System (non-user) programs perform those operations, and only those operations, which a user program *cannot* perform because of security and privacy requirements: a user is not permitted to interfere with or alter the programs or data of other users or the system, nor may he view data private to other users or groups of users.

Thus, for example, the searching, collating, and pattern matching aspects of information retrieval are performed by user programs having no special privileges. Of course, every user does not generate his own information retrieval programs, but it is possible for several rival retrievers to exist within the network. The system does the actual accessing of the secondary storage media and sees to it that the program handles only data belonging to, or shared by, the user running the program.

Only system operation is discussed here. One thing that sets the LLL computer system apart is that all system design and programming is done in house. Laboratory analysts and programmers created the network structure, the operating systems, the compilers, everything, from the ground up; commercial systems were not used even as a starting point. This enables the Laboratory to have a computer system that is better adapted to LLL needs and that incorporates the most modern concepts. For example, LLL is unique for having had interactive time-sharing on CDC 7600's since their original installation.

The present central data storage subnetwork uses a DEC PDP-10 as its concentrator. Its peripheral equipment includes CDC 844 disk packs, which hold the directories that act as the indexing mechanism to locate the central data files. A Librascope General Precision head-per-track disk is used as a buffer when files are transported among the worker computers and central storage. The central files themselves are kept on an IBM Data Cell and an IBM photodigital store. The entire central storage system is called Elephant, since an elephant never forgets.

The bulk of the centrally stored data is kept on the IBM photodigital store, which is about five years old. Only four others exist. One is at the Lawrence Berkeley Laboratory, one is at the Los Alamos Scientific Laboratory, and two are at the National Security Agency. Recording is on ordinary photographic silver halide film. The information to be recorded is supplied by the PDP-10 concentrator, but the detailed operation is managed by an IBM 1800 computer that is built into the photostore. The IBM 1800 operates all the mechanical features of the highly mechanical device, turning pneumatic switches, opening valves to release chemicals, and so on. It also senses the position of these switches and values to make sure that they actually operate; so if the machine starts to misbehave, the 1800 knows about it. After the film is written on with an electron gun, it is then put through a chemical developing process for about 2½ to 3 minutes. It emerges as a dry piece of developed photographic film and is stored in little plastic boxes called cells. These cells are moved around in the device pneumatically and are stored in compartmented trays. Two flying spot scanners are used to read the data.

A problem with photographic film is that one cannot verify that it has been manufactured flawlessly until one uses it. One cannot hold it up to the light to look for the flaws. So especially at the high density used in the photostore there is a considerable chance that the recording will have flaws in it. To overcome this problem, the IBM 1800 automatically generates an error detecting and correcting code that is put onto the film along with the actual data. Then during reading, if redundancy failures are detected, the device can attempt to regenerate the erroneous bits with the error-correcting facilities. It is possible for 10% of the data bits on a chip of film to be in error, and the errors will be detected and corrected.

Each chip of film is divided into 32 areas called frames, each of which holds 4100 36-bit words. There are 32 chips in a cell and 6750 cells in the machine. If one multiplies all these figures together, one finds a total exceeding one trillion (10^{12}) bits. If a recorded frame is viewed through a microscope, one can see the bits recorded as patterns of tiny black and white squares. A bit is recorded either as black followed by white or as white followed by black; it consists of two squares. So the average appearance of a line of data is gray, and an averaging mechanism in the film reader detects that. That is, at some level the reader obviously recognizes the individual bits, but at another level it knows that it is seeing an average gray as it scans along the lines. Between the lines of data there are alternately lines of solid black and lines of solid white. So if the flying spot drifts to one side it will begin to see an average shade that is too light, and if it drifts to the other side, it will begin to see a shade too dark. That is how the flying spot keeps itself on line, as in the old way of guiding aircraft.

The photostore has been in service about five years now. When it started out, of course, it had no data in it, and initially it filled up very slowly. The advent of the photostore might be compared to the invention of the banking system in the Middle Ages. At first people did not really believe that when one put one's money into the bank one would actually be able to get it back; so it took a while for banking to catch on. In the same way, LLL computer users were at first not sure that data could be put into the photostore and safely gotten back. The usage rose gradually, and it was not until September 1973, four years after it was put into service, that the photostore finally filled up with a trillion bits. There is a facility for manually taking cells out of the photostore and storing them on a shelf. At LLL the very oldest recorded information is removed and stored outside the machine so that newer data can be recorded. By May 1974, the oldest information in the machine was only

about 20 months old. The time for which the data remain on-line is shrinking, and if recording continues at the present rate there will be only about eight to ten months of information in the machine in mid-1975. So even a trillion bits of capacity is eventually not enough.

The system deals in units of information called *files*, which may include any number of data bits from zero up to a large maximum. In order to permit a user to very flexibly organize his collection of files and to make possible the most general sharing of files amoung users or groups of users, the central filing system is accessed through a system of *directories*. A directory is a body of information which is used to locate other bodies of information. In each directory is a number of *entries*. Each entry associates a name or mnemonic with a pointer or locator identifying another piece of information, which may be a file or a directory. A user identifies a file he wishes to access by giving the name associated with it in some directory. He in turn identifies that directory by giving a name associated with it in some other directory. For each user, this process begins with a unique directory associated with that user, called his *root* directory. The directories form a directed graph (tree-like) structure with the files as the terminal nodes. A user can access any directory or file which can be reached by a chain of entries starting in his root directory. The name of a file is associated with the way one accesses the file through a chain of directories, rather than with the file itself. A file or directory may be listed in more than one directory and have any number of different names. It is even permitted that there be closed loops such that one can start at a directory and follow a chain of pointers back to the starting place.

Directories permit the users to organize the structure of their files in any way they wish. The Elephant system deals only in files. The kind of activity that exists at the Laboratory does not require detailed access control to the record level. The system only makes sure that the files are delivered to the correct user, who is responsible for what he does with them. However, the system could be easily modified so as to have detailed control over the inner structure of a file, for example, by associating the file with a particular managing program which would mediate all accesses.

Each user of the system can access only those files that he can reach by starting in his root directory and following through a chain of directories until he gets to a file. Shared directories and files are those that can be reached from two or more root directories. Some files are shared simply because they are listed in two directories, but other files are shared, even if they are listed only once, because they are listed in a directory which is shared. The directory scheme conveniently and efficiently provides for the most general kind of sharing, in which any file can be shared by any subset of the users. Few, if any, other schemes could permit this when, as at LLL, the number of users exceeds 1000.

Consider, for example, a system with 1000 users. There are sets of files that only one user can get at, up to 1000 such sets. Then there are sets that two users can access, up to $1000 \times 999/2$ such sets, and so on. Altogether, sharing divides the files into $2^{1000} - 1$ sets, a rather horrendous number. Of course most of these sets would not actually occur, but the system would not know in advance which would occur. The directory structure solves the problem. If two users want to share some files, they either directly share them, or they set up a shared directory and put the files into them. Any number of users can be given access to a shared set of files, users working on the same project, for example. Of course there is a method, which will not be explained in detail, by which one user gives another user a pointer to a file or directory to initiate the sharing. Access to a file *via* a particular entry in a particular directory may be subject to inhibitions, such as by being read-only. This enhances the flexibility of sharing, permitting only some of the sharers of a file to modify it.

A file which is to be manipulated or edited by a user program is first copied from the central data base to the disk of a worker computer. The necessary operations are performed on this copy. At the completion of a session of activity, the altered file may be rerecorded on the central storage. Consider now how such file transport operations proceed. The user is running a program on one of the worker computers, text editing, information retrieving, compiling, or performing some other computation. For some reason he needs a file from central storage. His program sends a request out from the worker, through the network, to the Elephant system. The Elephant system queues the request (providing that it determines that the request is a legal request). When it is time to transport the file, Elephant contacts the worker computer that will be involved, and the file is transported (*i.e.*, copied to the worker computer's local storage). The worker tells Elephant whether it thought everything worked correctly, and Elephant takes that opinion together with its own and decides whether or not the operation was successful. Then it sends a reply to the user's program, informing it of success or failure. Elephant also can retry the job before it sends a final reply. The actual manipulation of the data is done on the copy of the data held on the disk of the worker computer. Giant storage devices like the photostore are unsuitable for frequent access. If a user wants to look at some words in a file, he should not ask for one word from the photostore, look at it, and later ask for another word. He should request that a good-sized chunk of the file (usually all of it) be copied to much more rapidly accessible disk storage, and he should work with it there. Then when he is done, he can send the updated copy back to the long-term storage. The photostore random-access time is about five seconds. That does not really matter if one moves the whole file at once and works with it locally, but to wait five seconds for every word or every few words would be intolerable.

Another comment about the photostore device is that one can only write on a chip of film once. Once the system has written on one of the chips of film, then the system is stuck with the chip until it is taken out of the machine together with all the other film that is in the same cell. (It is a very bad policy to have human beings open the cells and pick out separate chips of film, getting fingerprints all over them.) So this is a disadvantage: One cannot reuse the storage medium. Certainly the LLL photostore contains many files that nobody cares about any longer, but the film they occupy cannot be reused. On the other hand, there is a very important advantage to a nonrewritable medium: If one cannot rewrite it at all, then one cannot rewrite it by mistake. If one puts something onto the photostore, one can almost surely get it back. In fact there are only one or two instances of something put onto the photostore that cannot be recovered. These occurred because the mechanical mechanisms in the photostore literally crunched one of the chips of film.

A few remarks might be worthwhile about the LLL standard character set. It is a 256-character set with 128 of the characters being standard ASCII.[2] It was decided that, with the eight-bit byte apparently the definite trend in computer design, LLL should embed its character set in eight-bit bytes. The choice of ASCII seemed obvious because it is a better designed set than EBCDIC and because it is the American National Standard. Before selecting the additional 128 characters for the set, LLL sent a representative to the appropriate ANSI committee to seek guidance. So the LLL set is based on ANSI principles, but it is not standard, because there is no ANSI 256 character standard as yet.

Significant problems in the operation of the central storage facility are the following:

1. It is difficult to maintain sufficiently high average data

rates for transporting files among the worker computers and central storage. Although the disks used transfer at 10 to 40 MHz and the intercomputer channels can maintain a steady rate of over 7 MHz, the net average rate is currently about 1 MHz. All that transfer rate capacity is lost through friction: There are queues around the disks with one job waiting on another, and to transport something between two computers one must get a turn at a disk. One must get a core buffer. One must have the intercomputer channel available for one's use. One must get a core buffer and a turn at the disk in the computer at the other end of the channel. These facilities all must be available at the same time, but it is clear that some of them are going to have to wait while the other ones get ready. Delays also occur when a disk "slips" a revolution because it fills (or empties) a buffer faster than another (slower) disk or channel can empty (or fill) it. Although improved software algorithms may provide some relief, the only ultimate answer lies in hardware. What is needed are larger, cheaper buffer stores, greater multiplexing of interfaces, and faster interfaces.

One invention of value would be a two-headed disk, that is, a moving-head disk with two sets of moving heads and two controllers. One could write on the disk with one set of heads, which would march sequentially across the tracks of the disk so that there would never be any significant head-motion delays, and one would read with the other set of heads, which would also march sequentially, but behind the writing set. Essentially one would have a large elastic store into which data could be dumped out of one computer and held until the receiving computer was ready.

2. Reliability is less than ideal. Actually availability is a better word. The system tends not to lose data nor to get it wrong. That is, once one has stored something, one can usually get it back. But one does not always get it at the moment one wants it. In a system as large as the Octopus network, with all its many components, it is difficult to keep all the components in simultaneous working condition. Only about 80% of the time is the whole system working to the point that a user wanting to get a file from the central filing system can actually get it at that moment. The worker computer he is using might be having a bad time, or the interfaces between the different links in the network might be, or one of concentrators, etc. The one thing that gives the most trouble is the photostore itself. It is a very Rube Goldberg device, and it is perhaps remarkable that it works at all. On the average of about four times a day, it will unpredictable stop working properly, the maintenance man will leap into action, and it will probably be back on the air within five or ten minutes. All the users that wanted a piece of information during that five or ten minutes are unhappy, and there is not much one can do about it. The solution to the reliability/availability problem lies in better hardware, more tolerant software, redundant interconnections, and sophisticated diagnostics.

3. Capacity is less than is required. Even a trillion bits eventually runs out, particularly when utilized by over 1000 users, seven days a week, for five years, through the action of programs executing at over one million instructions per second. It has been found to be difficult to determine in advance which data sent to the photostore are actually going to have little long-term use and which should remain on-line essentially forever. Once two items are recorded onto the same chip of film, they of course must both stay in the device or both be moved to the shelf together. Next year a new, rewritable medium of over half a trillion bits will be added to the network in an effort to relieve the capacity problem.

In spite of these problems, the Octopus central storage system is a useful and valuable facility which greatly increases the productivity of LLL computer users.

LITERATURE CITED

(1) Fletcher, J. G., "The Octopus Computer Network," *Datamation*, **19** (4), 58–63 (April, 1973).

(2) Fletcher, J. G., "Characters in a Dialog," *Datamation*, **20** (10), 42–47 (October, 1974).