USING A SMALL/LOW COST COMPUTER IN AN INFORMATION CENTER*

Daniel U. Wilde

New England Research Applications Center
University of Connecticut
Storrs, Connecticut

Abstract

In the past, mechanization has been limited to those information centers that have access to extensive computer facilities. Now, small/low cost computers are available with I/O capacities that make them suitable for SDI and retrospective searching on any of the many commercially available data bases. A small two-tape computer system is assumed, and an analysis of its run-time equations leads to a three-step search procedure. Run times and costs are shown as a function of file size, number of search terms, and input transmission rates. Actual examples verify that it is economically feasible for an information center to consider its own small, dedicated computer system.

---

* This research was sponsored in part by the National Aeronautics and Space Administration Contract NASW - 2307.

Introduction

Until recently, information center mechanization has been limited to those centers that have access to extensive computer facilities. Although the amount of machine time that a center uses may be small, the center is dependent upon the services of highly specialized computer personnel. Consequently, computerization has occurred in large centers that can afford their own computer system or those centers that have convenient access to someone else's machine.

By now, the small computer has gained wide acceptance and its use for special applications has become commonplace. For example, Warheit (1) suggests the IBM System/3 as a small, general purpose computer which can perform a great variety of library jobs, such as controlling book circulation and producing overdue notices. Furthermore, he feels that it is cheap enough so that many libraries which formerly could not afford computer services now have the opportunity to have their own system (2).

IBM has recently announced that inexpensive tape drives will soon be available for its System/3. This means that this machine can be used by information centers to process externally produced data bases that are distributed on tape, such as the Library of Congress MARC file. In addition, it will be possible to use this system for SDI and retrospective searching on any of the many commercially available data bases. Let us now consider how this might be done with a small/low cost computer.

The Computer System

At this point, let us assume that the computer budget is limited and that hardware costs are to be kept low. Therefore, let us begin with the small system described by Warheit and add two tape drives. Now, we have a machine that includes a small CPU with limited memory, a card input/output device, a printer, and two tape drives. This minimal configuration in a System/3 will rent for approximately $2,600.00 to $2,900.00 per month when tapes are available. This same configuration is currently available in an IBM 1130 and rents for approximately $2,700.00 to $3,400.00 per month. Both include a small disk for storage of a monitor and user programs.

Search Procedure

With such a small machine, random access searching on inverted files is out of the question. This is particularly true for large data bases, such as the American Society for Metals (ASM) file of 125,000 documents or the National Aeronautics and Space Administration (NASA) file of 650,000. Furthermore, file inversion on such a limited machine would be relatively difficult since the system includes just two tapes and one small disk. Consequently, both search and file maintenance procedures dictate that data files be organized in a linear mode. This requires that each and every document be examined during each search. Needless to say, this may result in long search runs.

Appendix 1 shows equations that can be used to predict linear search times. Analysis of these equations shows that run time is a function of three sets of parameters. The first set is dictated by the data file and includes the number of data bank documents and the average number of index terms per document. The second set is determined by the computer and includes average instruction execution time, start time of the input tape drive, and time to transmit one input character from the drive to memory. The third set is specified by the information center and includes the average number of characters per input document and the number of search terms per retrieval run. Let us now consider how these last two parameters can be varied by the information center so as to lessen search times.

The average number of input characters can be reduced by eliminating non-essential information from the data file. Here, document records are edited to conform to the needs and demands of center users. On a small machine with limited memory, this reformating may require multiple passes. Here, the output tape is used for storage of intermediate results which become the input to the next pass.

The number of search terms can be reduced by examining the construction of search strategies. For simplicity, let us assume Boolean logic. From the equation

$$S = A * (B + C) + D * E$$

it can be seen that if S is to be true, then one of the following abbreviated equations, S_i, must also be true.

$$S_1 = A + D \qquad S_2 = A + E$$

$$S_3 = B + C + D \qquad S_4 = B + C + E$$

If for a given document an S_i is false, then S is also false. Therefore, S need only be evaluated for those documents for which S_i is true.

This approach leads to a three-step retrieval procedure. In the first step, the complete file is searched using an abbreviated strategy, S_i. Here, documents are read from the data file using one tape drive as input. When a document is found that satisfies S_i, it is copied onto the other tape. If a document does not satisfy S_i, it is skipped since it cannot fulfill S. After the complete data file has been scanned, the output file contains the subset of documents that satisfies S_i. During the second step, this sub-file is searched using the complete strategy, S. When a document is found that satisfies S, it is copied onto the output file along with a tag that identifies its requestor. Finally, in the third step, search results are printed. Here, documents are grouped together by requestor by passing the print tape once for each request.

Generally, run time for the first step is minimized by selecting an abbreviated strategy which contains the least number of terms. For the S shown above, this implies S_1 or S_2. Second-step run time is minimized by selecting the abbreviated strategy which produces the smallest output during the first step. Here, to make a final choice, we must know the relative occurrences of terms: A, D, and E. Lastly, print time during the third step depends upon the number of documents retrieved by S in the second step.

If several searches are batched together, the abbreviated equation for

the first step is simply the sum of the individual abbreviated equations. On the other hand, abbreviated strategies can be selected so as to permit the strategy designer to iteratively partition his data base into sub-files and to quickly examine sub-file content as described in Wilde (3). In contrast, abbreviated strategies can be designed such that several complete strategies can be run in parallel during the second step as suggested by Junkins and Schultz (4).

Search Times

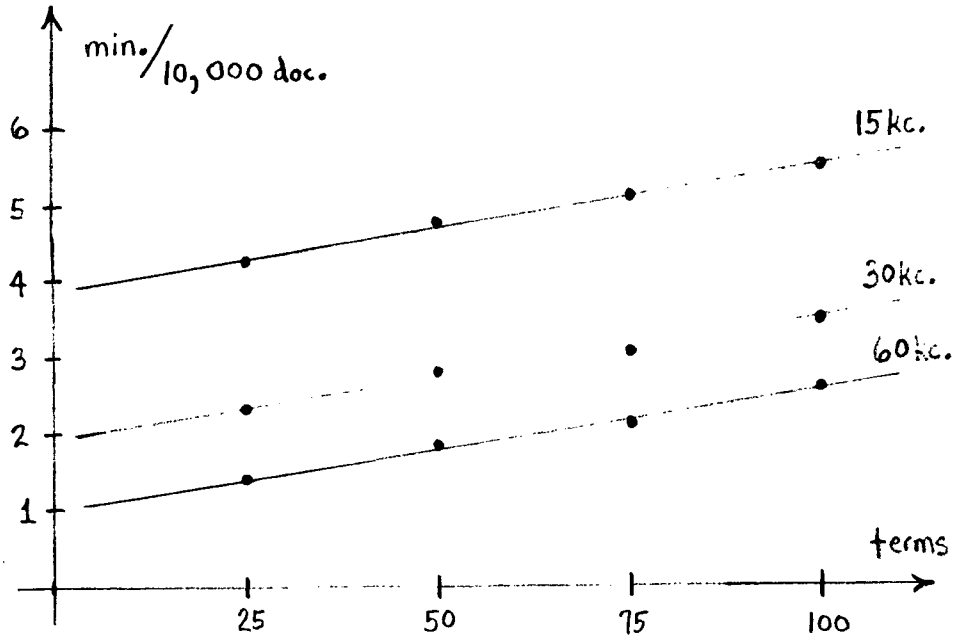
The equations in Appendix 1 predict that search time grows linearly as the number of search terms increases. This is confirmed by experimental data displayed in Figure 1. Here, run time per 10,000 documents is shown for three different input transmission rates as a function of the number of abbreviated strategy terms. These graphs can be used to estimate run times for different system configurations and data file sizes. For example, an SDI search of 25 abbreviated terms on an update of 5,000 documents using a 15kc input drive takes approximately

$$\frac{5{,}000 \text{ doc.}}{10{,}000 \text{ doc.}} \cdot 4.26 \text{ min.} \approx 2.1 \text{ min.}$$

Similarly, a retrospective search of 10 abbreviated terms on a file of 650,000 documents using a 60kc drive takes approximately

$$\frac{650{,}000 \text{ doc.}}{10{,}000 \text{ doc.}} \cdot 1.14 \text{ min.} \approx 75 \text{ min.}$$

Figure 1 - Run Time versus Search Terms (IBM 1130 - No overlap)



Search costs can be estimated by multiplying predicted run times by computer rates. Table 1 presents a set of monthly and hourly rates for a two-tape IBM 1130 system using the same three input speeds shown in Figure 1. (Hourly rates assume 176 hours/month.) Here, rates are given for a total system and for tapes alone. This latter figure would be appropriate if a non-tape system were already installed and if the extra tape costs were of interest.

Using the same examples as above, the system cost for the SDI would be

$$\frac{\$17.90/\text{hr.}}{60 \text{ min/hr.}} \cdot 2.1 \text{ min.} \approx \$.65$$

while the tape-only cost for the retrospective would be

$$\frac{\$10.94/\text{hr.}}{60 \text{ min/hr.}} \cdot 75 \text{ min.} \approx \$13.70$$

Summary

If an information center is to be successful, it must be responsive to the demands of its users and clients. If a center has its own computer system, it can schedule batched runs, special runs, or evening runs in order to satisfy client demands, to meet higher priorities, or to overcome equipment failures. When a center has its own machine, it is paying a flat rental fee or fixed monthly amortization charge. Thus, additional computer use results in a lower per unit cost. Until recently, computer systems with good I/O were too expensive for most centers. Now, small/low cost machines are available that permit a center to consider acquiring its own dedicated computer system.

Acknowledgement

I would like to thank Mr. Stuart Harris for his aid in programming the NERAC search system and Mrs. Susan Abramson for her help in generating the test data.

Appendix 1 - Search Time Equations

Total run time, T_{run}, for a linear search is a function of total central processor time, T_{cpu}, and total tape time, T_{tape}. If computer operations are overlapped, T_{run} equals the larger of T_{cpu} and T_{tape}. If they are not, T_{run} equals the sum of T_{cpu} and T_{tape}.

Total tape time, T_{tape}, is a function of total input time, T_{input}, and total output time, T_{output}. Generally, since the number of retrieved documents is much less than the number of data bank documents, T_{output} is very small relative to T_{input}. Therefore, for simplicity, T_{output} is ignored.

T_{input} is the sum of the time spent starting the input tape drive, T_{start}, the time transmitting information from the drive to memory, T_{trans}, and the time stopping the drive, T_{stop}. Even the most elementary tape drive sends an end-of-transmission signal at each inter-record gap; and thus, there is no need to wait for the drive to stop. Therefore,

$$T_{tape} = T_{start} + T_{trans} \tag{1}$$

The total time spent starting the input drive is the product of the time per start, t_{start}, and the number of starts during a search. If the file being searched contains N_{doc} and if the input tape is blocked at an average of n_{block} documents per block, then total start time is

$$T_{start} = \frac{N_{doc}}{n_{block}} \cdot t_{start} \tag{2}$$

The total tape transmission time is the product of the number of documents, N_{doc}, the average number of characters transmitted per document, n_{char}, and the time to transmit each character, t_{char}. Therefore, total transmission time is

$$T_{trans} = N_{doc} \cdot n_{char} \cdot t_{char} \tag{3}$$

Total tape time is then found by substituting (2) and (3) into (1) producing

$$T_{tape} = N_{doc} \left\{ \frac{t_{start}}{n_{block}} + n_{char} \cdot t_{char} \right\} \tag{4}$$

The total central processing time, T_{cpu}, can be expressed as the product of the number of documents, N_{doc}, and the average time to process each document, t_{doc}.

$$T_{cpu} = N_{doc} \cdot t_{doc} \tag{5}$$

Here, the average time to process each document is the product of the average number of instructions to process that document, n_{inst}, times the time to execute an average instruction, t_{inst}.

$$t_{doc} = n_{inst} \cdot t_{inst} \tag{6}$$

In processing a linear file, document terms must be compared against question terms. The algorithm chosen to perform these comparisons must be a function of the average number of index terms per document, i_{doc},

and the number of retrieval search terms, i_{search}. If each document term is compared against each search term, then the average number of instructions to process each document is

$$n_{inst} = i_{doc} \cdot i_{search} \cdot n_{comp} + n_{house} \tag{7}$$

where n_{comp} specifies the number of instructions to make a term comparison while n_{house} represents the instructions to perform housekeeping functions. If both search and document terms have been previously sorted, this expression can be improved to

$$n_{inst} = (i_{doc} + i_{search}) \cdot n_{comp} + n_{house} \tag{8}$$

Substituting (6) and (8) into (5), total CPU time becomes

$$T_{cpu} = N_{doc} \cdot \left\{ (i_{doc} + i_{search}) \cdot n_{comp} + n_{house} \right\} \cdot t_{inst} \tag{9}$$

Examination of equations (4) and (9) shows that run time is a function of three independent sets of constraints. The first set is dictated by the data file and includes the number of data bank documents and the average number of index terms per document. The second set is determined by the computer and includes average instruction execution time, start time of the input tape drive, and time to transmit one input character from the drive to memory. The third set is specified by the information center and includes the average number of documents per input block, the average number of characters per input document, and the number of search terms per retrieval run. Here, the number of documents per block should be adjusted so as to produce a balanced run where tape time and CPU time are nearly equal. Balancing is discussed in Gildersleeve (5).

References

1. Warheit, I.A., The Small Computer and the Library, Proceedings of the American Society for Information Science, Volume 7, 1970, pp. 91-93.

2. _____, Library Automation: The IBM System/3, IBM Corporation, In Press.

3. Wilde, D.U., Iterative Strategy Design, American Documentation, January, 1969, pp. 90-91.

4. Junkins, K. and L. Schultz, An Alternate Strategy to Iterative Searching, Proceedings of the American Society for Information Science, Volume 7, 1970, pp. 323-325.

5. Gildersleeve, T.R., Design of Sequential File Systems, Wiley - Interscience, New York, 1971.