METHODS OF RANDOMIZATION OF LARGE FILES WITH HIGH VOLATILITY

Patrick C. MITCHELL: Senior Programmer, Washington State University, Pullman, Washington, and
Thomas K. BURGESS: Project Manager, Institute of Library Research, University of California, Los Angeles, California

Key-to-address conversion algorithms which have been used for a large, direct access file are compared with respect to record density and access time. Cumulative distribution functions are plotted to demonstrate the distribution of addresses generated by each method. The long-standing practice of counting address collisions is shown to be less valuable in judging algorithm effectiveness than considering the maximum number of contiguously occupied file locations.

The random access disk file used by the Washington State University Library Acquisition sub-system is a large file with a sizable number of records being added and deleted daily. This file represents not only materials on order by the Acquisitions Section, but all materials which are in process within the Technical Services area of the Library. The size of the file currently varies from approximately 12,000 to 15,000 items and has a capacity of 18,000 items. Over 40,000 items are added and purged annually. Each record consists of both fixed length fields and variable length fields. Fixed fields primarily contain quantity and accounting information; the variable length fields represent bibliographic data. Records are blocked at 1,000 characters for file structuring purposes; however the variable length information is treated as strings of characters with delimiters. The key to the file is a 16-character structure which is developed from the purchase order number. The structure of the key is as follows: six digits of the original purchase order number, two digits of partial order and credit information, and eight digits containing the computed relative record address. Proper development of this key turns out to be

the most important factor in achieving efficiency in both file access time and record density within the file.

The W.S.U. purchase order numbering system, developed from a basic six-digit purchase order number, allows up to one million entries. Of these, the Library currently uses four blocks: one block for standing orders, one block for orders originating from the University after the system becomes operational, another block used by the systems people in prototype testing of the system, and a fourth block which was given to one vendor who operates an approval book program.

In mapping a possible million numbers into eighteen thousand disk locations, there is a high probability that the disk addresses for more than one record will be the same. Disk location, also called disk address, home position, and relative record address (RRA) in this paper, refers to the computed offset address of a record in the file, relative to the starting address of the file. Currently, the file resides on an IBM 2316 disk pack which can store six 1000-character records per track. Thus if the starting address of the file is track 40, a record with RRA = 5 would have its home position on track 40, while a record with RRA = 6 would have its home position on track 41. It should be noted that routines in this system are required to calculate neither absolute track address nor relative track address and therefore the file could be moved to any direct access device supported by OS/BDAM without program modification.

When two records map into the same address, it is called a collision. For a WRITE statement under the IBM 360 Operating System, Basic Direct Access Methods, the system locates that disk address generated and if another record is found there, it sequentially searches from that point forward until a vacant space is found and then stores the new record in that space. The sequential search is done by a hardware program in the I/O channel and proceeds at the rotational speed of the device on which the file resides. The CPU is free during this period to service other users. Similarily, when searching for a record, the system locates the disk address and matches keys; if they do not match, it sequentially searches forward from that point. Long sequential searches sharply degrade the operating efficiency of on-line systems.

In initial experimentation with this file, it was discovered that some records were 2,500 disk positions away from their computed locations. This seriously reduced response time to the terminals which were operating against those records. The necessity to develop a method for placing each record close to its calculated location became quite obvious. However, the methodology for doing this was not as clear.

The upper bound delay for a direct access read/write operation can be defined as the largest number of contiguously occupied record locations within the file. The problem of minimizing this upper bound for a particular file is equivalent to finding an algorithm which maps the keys in such a way that unoccupied locations are interspersed throughout the

file space. One method for doing this is to triple the amount of space required for the file. This has been a traditional approach but is unsatisfactory in terms of its efficiency in space utilization.

The method first used by the Library was motivated by the necessity to "get on the air." Its requirements were that it be easily implemented and perform to a reasonable degree. The prime modulo scheme seemed to qualify and was selected. As this algorithm was used, the largest prime number within the file size was divided into the purchase order number and the modulo remainder was used as an address; that is, RRA = [Po Modulo Pr] where RRA is the relative record address, Po is the Purchase Order Number, and Pr is a prime number. During the initial period file size grew to about 8,000 records. Because the Acquisitions Section was converting from its manual operation, the file continued to grow in size and the collision problem became pronounced. When the file reached about 70% capacity—that is when 70% of the space allocated for the file was being occupied by records—this method became unusable; records were then located so far from their original addresses that terminal response times became degraded and batch process routines began to have significant increases in run times.

With no additional space available to expand the size of the file, it became necessary to increase the record density within the existing file bounds. Therefore an adaptation of the original algorithm was developed. In addition to generating the original number by dividing a prime number into the purchase order number and keeping the modulo remainder, the purchase order number was multiplied by 300 and divided by that same prime number to get an additional modulo remainder; the latter was added to the first modulo remainder and the sum then divided by 2:

$$\text{RRA} = \frac{(\text{Po Modulo Pr}) + (300 \bullet \text{Po Modulo Pr})}{2}$$

Again this scheme brought some relief, but the file continued to grow as the system was implemented, and it became obvious that this procedure would also fail because of over-crowded areas in the file.

A search of the literature using W. B. Climenson's chapter on file structure (2) as a start provided some other methods for reducing the collision problem (1, 3, 4, 5, 6). Several randomization or hashing schemes were examined. However, none of these methods appeared to be particularly pertinent to the set of conditions at Washington State.

In order to bring relief from the continuing problem of file and program maintenance involved with changing the file-mapping algorithm, research was initiated to devise an algorithm which would, independent of the input data, map records uniformly across the available file space.

The algorithm which resulted utilizes a pseudo-random number generator, RAND (7) developed at the W.S.U. Computing Center RANDL, Program 360L-13.5.004, Computing Center Library, Computing Center,

Washington State University, Pullman, Washington. The normal use of RAND is to generate a sequence of uniformly distributed integers over the interval [1, M], where M is a specified upper bound in the interval [1, $2^{31} - 1$]. In addition to M, RAND has a second input parameter: N, which is the last number generated by RAND. Given M and N, RAND generates a result R. RAND is used by the algorithm to generate relative disk addresses by setting M to the size or capacity of the file, by setting N to the purchase order number of the record to be located, and by using R as the relative address of the record. RRA = RAND (Po, M).

In order to test the effectiveness of this algorithm and others which might be devised, a file simulation program was written BDAMSIM, Program 360L-06.7.008, Computing Center Library, Computing Center, Washington State University, Pullman, Washington. Inputs to this program are: a) an algorithm to generate relative record locations; b) a sequential file which contains the input data for "a"; c) various scalar values such as file capacity, approximate number of records in the file, title of output, etc.

The program analyzes the numbers generated by "a" operating on "b" within the constraints of "c". The outputs of the program are some statistical results and a graphical plot showing the cumulative distribution function of the generated addresses.

Figures 1, 2, and 3 show the plotted output of the three algorithms operating against the current acquisitions file. The abscissas of the plots

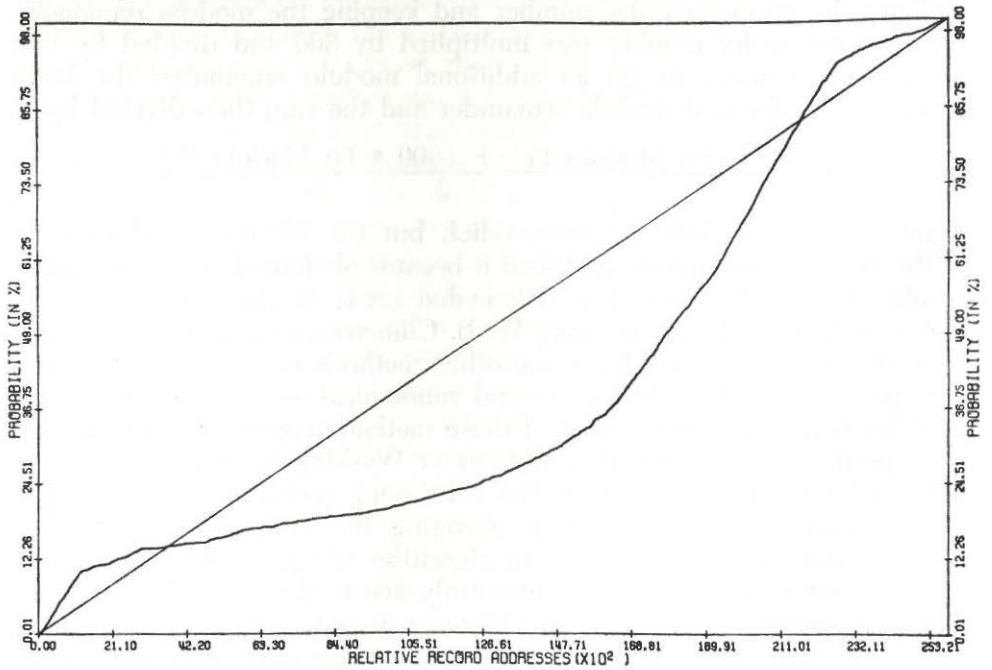

Fig. 1. RRA = Po Modulo Pr

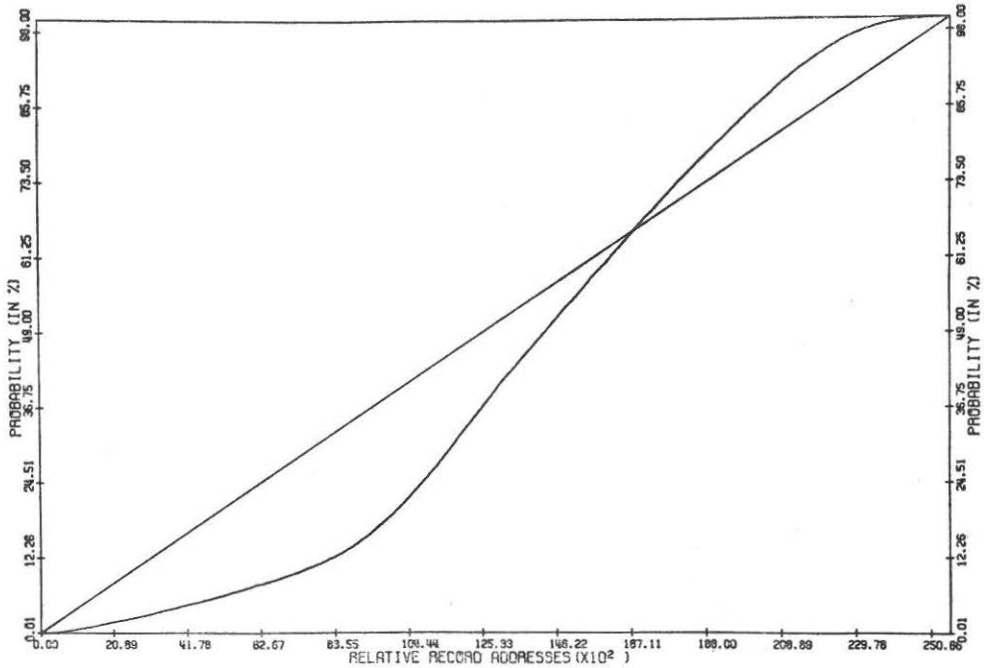

Fig. 2. RRA = ((Po Modulo Pr) + (300 × Po Modulo Pr))/2.

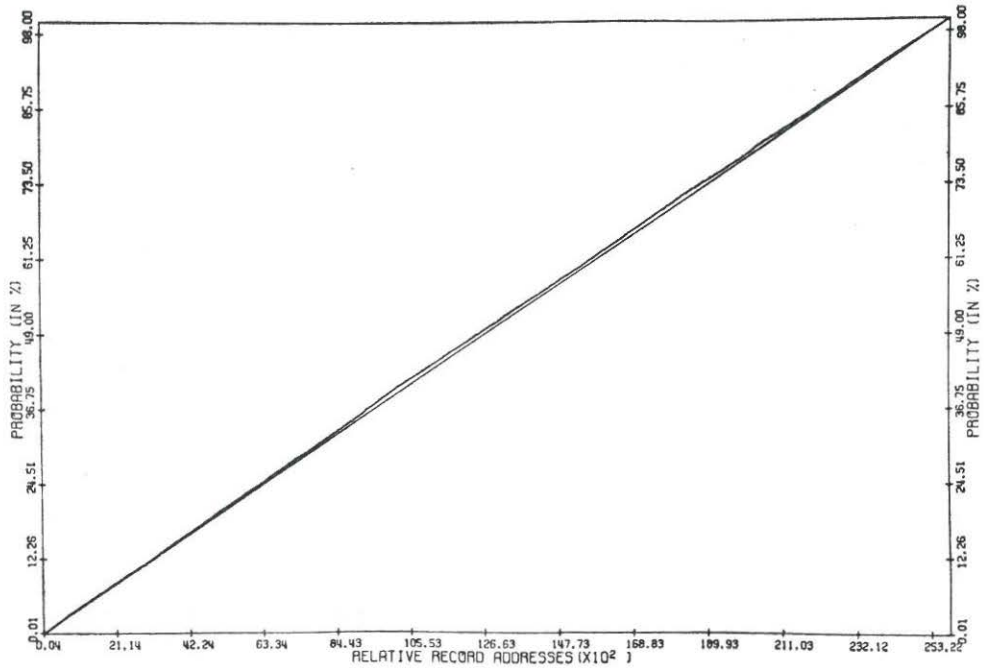

Fig. 3. RRA = RAND (Po, Pr).

represent disk addresses which were generated. The ordinates represent the probability of generating disk addresses less than or equal to addresses on the abscissas. The ideal algorithm will uniformly distribute the records throughout the file and its plot will be a straight line from the lower left corner of the plot to the upper right corner. The density of the file is represented by the slope of the plot. A steep (vertical) slope represents many records clustered together; a mild (horizontal) slope represents a less densely populated area; a slope of 1 represents a uniform distribution.

In Table 1 the three mapping methods are compared with respect to the number of collisions and the upper-bound read/write delay. These statistics are based on the current acquisitions file at 90% capacity (12,913 records and 14,347 locations). Looking at only the collision statistics, it would appear that method I, RRA = [Po Mod Pr], would provide the best mapping. However, its upper-bound read/write delay indicates that one area of 8,823 locations is contiguously occupied. Any new record to be inserted near the beginning of this area would necessarily be stored over 8,800 locations away from its computed address. Method II would be a less effective algorithm for use with the current acquisitions file, as it has a larger (11,585) upper bound read/write delay than method I and has fewer records in their home positions. When method II was implemented, it represented an improvement over method I. Since that time, the data base contents have changed and so have the comparative performance statistics of the methods. This illustrates the dependence of the effectiveness of these methods upon the data base itself. The statistics for method III, RRA = [RAND (Po, Pr)], indicate that more records are located a short distance from their computed addresses than in method I. However, no record is more than 338 locations away.

TABLE 1. *Comparison of Three Mapping Methods*

	RRA	*Percent of Records Having 0-6 Collisions*							*Upper Bound Read/Write Delay*
		0	1	2	3	4	5	6	
I	Po Mod Pr	44.24	42.48	11.04	1.95	.23	.05	—	8,823
II	(Po Mod Pr + 300 Po Mod Pr)/2	31.31	40.44	19.56	6.72	1.66	.19	.11	11,585
III	RAND (Po, Pr)	37.02	35.34	19.61	6.70	1.08	.23	.11	388

Table 1 can be understood by considering a simple example. Suppose there are six records to map into a direct access file of eight locations. Suppose that RRA generator A assigns home positions {2, 4, 3, 4, 3, 2} to the records and RRA generator B assigns home positions of {0, 4, 0, 4, 4, 0}. If E denotes an empty, or non-occupied position in the file, the corresponding file maps would actually appear as:

	0	1	2	3	4	5	6	7
A	E	E	2	3	4	4	3	2
B	0	0	0	E	4	4	4	E

On the basis of Table 1, and with X representing the number of collisions, the statistics would be as follows:

	X = 1	X = 2	UBRWD
A	100	—	6
B	—	100	3

While method A provides fewer collisions, it has saturated one area of the file space, indicated by the high UBRWD. Method B provides a superior mapping in that file saturation has been minimized.

The acquisitions file is contained on an IBM 2316 disk pack. It has been found that a read/write delay of 200 locations is about equivalent to one second. Hence method I represents a maximum delay of 44 seconds, method II a maximum delay of 58 seconds, and method III a maximum delay of 2 seconds. The Library systems group, together with the Library staff, have arbitrarily set the maximum delay for terminal response time at 2½ seconds. Utilizing methods I or II, this limit was reached when the file was 60-80% full. With method III, the limit is not reached until the file is 90-95% full.

From these data it can be seen that the method utilizing the random number generator is clearly an improvement over the other methods tested. This simulation has been performed against the acquisitions file at regular intervals to see if it would continue to be uniformly populated. It has been found that even under highly volatile conditions, the records in the file remain uniformly distributed throughout the available file space.

REFERENCES

1. Burgess, T. K.; Ames, J. L.: *LOLA, Library On-Line Acquisition Sub-System,* (Washington State University Library, 1968).
2. Climenson, W. D.: "File Organization and Search Techniques," *Annual Review of Information Science and Technology,* 1 (New York: Interscience, 1966), 107-135.

3. Hanan, M.; Palermo, F. P.: "An Application of Coding Theory to a File Address Problem," *IBM Journal of Research and Development,* 7 (April, 1963), 127-129.
4. Hayes, R. M.: "A Theory for File Organization," *On-Line Computing,* (New York: McGraw-Hill, 1967), pp. 264-289.
5. Kaimann, R. A.: "Entry to the File, Randomize or Index," *Data Processing Magazine,* 10 (December, 1968), 24-27.
6. Lefkovitz, David: *File Structures for On-Line Systems,* (New York: Spartan, 1969).
7. Payne, W. H., et. al.: "Coding the Lehmer Pseudo-Random Number Generator," *Communications of the ACM,* 12 (February 1969), 85-86.