Incremental Cost of Encoding Data in a Tape-Based Information Storage and Retrieval System

or

It's Dirt Cheap to Scramble Your Data!

Up to this point, no published results have appeared dealing with the cost of encoding (scrambling) information in an information storage and retrieval system. Opponents of more stringent access controls in computer systems have cited the high cost of adding additional software control functions to the already overburdened operating systems, but to this author's knowledge no hard timing data has yet appeared.

In a quest for some ballpark figures on the costs of encoding data, (and conversely the costs of decoding data), an experiment was run on the IBM 360/67 computer system at the Campus Facility of Stanford University Computation Center. A tape containing 1,050 80-character card images in clear (unscrambled) format was generated and the 80-character records were sequentially read in, scrambled, and the (encoded) card images written out on a new output tape. The appendix contains the listing of the FORTRAN program used to do this job, and also the printout of the timing results.

As can be seen by an examination of the printout, the time used to scramble 1,050 cards using the simple encoding technique shown in the program listing, was 7,562 seconds. (Actually this is the time used to scramble 1,049 cards, since writing of the first output record required a one-time opening of the data set by the operating system. Since we were trying to get an estimate of the time required to encode a large number of records, this "open time" was disregarded. It would certainly be negligible for any
large-scale data base.) This time, 7,5612 seconds, is the total wall-clock time used from the time the first clear record was read in until the time the last encoded record was written out onto the output tape. All waits for input/output, etc., are included in this time. Again, the time does not represent central processor cycles. It is wall-clock time on a system where this was the only job running in addition to the operating system. (In other words, this job used the entire 600K byte partition on the 360/67.) The experiment was carried out in this manner in order to get a high estimate of the incremental cost involved in scrambling a large number of cards.

In a multi-programming system the actual time used in encoding could be overlapped with tasks from other jobs and therefore would not nearly be so costly.

In this worst case we see that 1,049 cards were scrambled in 7,5612 seconds. We can put it another way; the incremental cost of encoding one card image on this system is .007 seconds. Under the existing rate structure at the Stanford Computation Center, it then costs 0.08 cents to encode each card image. Therefore, encoding one card image (80 bytes of information) for each of the 20,000,000 residents of the State of California would cost only $16,000.

Conclusion

The incremental cost of scrambling information in a large computer data base seems infinitesimal.
Appendix. Printout from Experiment, Including FORTRAN Code and Results
COMPILED OPTIONS = NAME = MAIN,INPUT=GO, LINES=05, ST, SOURCE, EBCDIC, N\LIST, NZ, DECK, LOAD

002 BLOCK DATA
003 INCLUDE  'IUSTAT.I'
004 COMMON  'CONSTANTS', 'HUGS', 'NUM ', 'MAX USERS', 'MAX-LIST', 'ITALK',
005 1 FORM, 2 FORM2, 3 FORM3,
006 4 FORM4, 5 FORM5, 6 FORM6,
007 7 FORM7, 8 FORM8,
008 9 FORM9, 10 FORM10,
009 11 FORM11, 12 FORM12,
010 13 FORM13, 14 FORM14

011 DATA IHRND+/ 14635673, 12463133, -999999471, 107694648,
012 DATA 15671432, 137371369, -754517039, 227935190,
013 DATA 24995190, -25546193, 12594180, 13331762,
014 DATA 12694095, -143295037, 243176905, -240111797,
015 DATA 139833006, -176943561, -219461227, 176003683

016 DATA USER /1325/

017 DATA PASSW/1325/

018 DATA USER CONTROL BLOCK (TEMPLATE)

019 INTEGER IHRND(+), LUSER(10), PLTOL(10), UTL(10)

020 C USER CONTROL BLOCKS -- MAX USR=10, NUCB=3

021 INTEGER IHRND(1), LUSER(10), PLTOL(10), UTL(10)

022 C USER IDENTIFICATIONS TRANSFORMED BY APPROPRIATE ALGORITHM

023 DATA UTL(13)/138464165, 1803822108, -1730839952,
024 DATA /-154787608, 640,
025 DATA 193529795, 203596489, -116448726,
026 DATA /-126850271, 640,
027 DATA 169257097, -50119502, 122354686, 740,
028 DATA 169257097, -1377685127, 1618028494, 101222, 640,
029 DATA 640,

030 C PASSWORDS TRANSFORMED BY APPROPRIATE ALGORITHM

031 DATA PWTL(13)/1691037135, -118253089, -591364798,
032 DATA /-598895777, 640,
033 DATA -126090104, 751179207, -1043920360, -509549044,
034 DATA 101222, 540,
035 DATA 1860945598, 1801809415, -793937047, 740,
036 DATA -192604301, 2064559437, -2039533773, 814026026,
037 DATA 643, 5600,

038 C CONSTANTS USED THROUGHOUT THE SYSTEM

039 DATA ITALK/2/

040 C LENGTH OF ARRAY PASSED BY TALK PROGRAM TO ACCESS PROGRAM

041 DATA MAXLIST/100/

042 C MAXIMUM LENGTH OF LIST OF LUCKED DATUMS MAINTAINED BY ACCESS PGM

043 DATA USR(11)/-2/

044 C INITIALIZE TO NO ACTVE USER CONTROL BLOCKS

045 C INITIALIZE TO CRITICAL SECTION OF ACCESS PROC. NOT CURRENTLY IN USE

046 DATA NUCH/3/
<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>C WORD</td>
<td>No. of Words in Each User Control Block</td>
</tr>
<tr>
<td>C USERSE / MAXSE</td>
<td>Max. No. of User/Terminal Combinations</td>
</tr>
<tr>
<td>C POSSIBLE AT ANY GIVEN TIME</td>
<td></td>
</tr>
<tr>
<td>DATA FROM</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 1</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 2</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 3</td>
<td></td>
</tr>
<tr>
<td>C INTERNAL NAMES OF FORMULARIES</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 4</td>
<td></td>
</tr>
<tr>
<td>C LIST OF FORMULARIES CURRENTLY IN THE SYSTEM</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 5</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 6</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 7</td>
<td></td>
</tr>
<tr>
<td>C INTERNAL NAMES</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 8</td>
<td></td>
</tr>
<tr>
<td>C (IN THIS CASE ALSO VIRTUAL ADDRESSES)</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 9</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 10</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 11</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 12</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 13</td>
<td></td>
</tr>
<tr>
<td>C POSSIBLE AT ANY GIVEN TIME</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 14</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 15</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 16</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 17</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 18</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 19</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 20</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 21</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 22</td>
<td></td>
</tr>
<tr>
<td>C WORD</td>
<td></td>
</tr>
<tr>
<td>C USERSE / MAXSE</td>
<td></td>
</tr>
<tr>
<td>C POSSIBLE AT ANY GIVEN TIME</td>
<td></td>
</tr>
<tr>
<td>DATA FROM</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 1</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 2</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 3</td>
<td></td>
</tr>
<tr>
<td>C INTERNAL NAMES OF FORMULARIES</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 4</td>
<td></td>
</tr>
<tr>
<td>C LIST OF FORMULARIES CURRENTLY IN THE SYSTEM</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 5</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 6</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 7</td>
<td></td>
</tr>
<tr>
<td>C INTERNAL NAMES</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 8</td>
<td></td>
</tr>
<tr>
<td>C (IN THIS CASE ALSO VIRTUAL ADDRESSES)</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 9</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 10</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 11</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 12</td>
<td></td>
</tr>
<tr>
<td>DATA FROM 13</td>
<td></td>
</tr>
<tr>
<td>C POSSIBLE AT ANY GIVEN TIME</td>
<td></td>
</tr>
</tbody>
</table>
COMMON/CONSTANTS, INFO, MAX USERS, MAX LIST, LIST,}
1 ERR, ERR3, ERR3,
2 NEXTAIL, SAMBAI,
1 FETCH, STORE, UNLEP, UNLSTP, FSHLP, SAMLPT, ATACHP, DETACHP
LITCEN, SKRNFDPY, CLRPUL, LCLRUF(4), IXL BNL(4)
LITCRIP/DMDT/IRAND/ DPT,PassWD, USRN, CARD, UPLTR, IEPTUL, IUIUL
LITCERP, PASSWD(1), USRNL, UPTUL(10,10), UTRUL(10,10), UPTUL(10,10)
LITCRIP RPT(4), CAMNL(160), IRLND(10)

**CERTIFIED 8 MAY 1960**

**FORMULARY SELECTOR**

**GO TO (3,1,3), II**

**IF (IUCR(3)**

**GO TO (3,1,3), III**

**IF (IUCR(3)**

**GO TO (3,1,3), IV**

**IF (IUCR(3)**

**GO TO (3,1,3), V**
C THIS PROGRAM SCRAMBLES THE DATA IN A DATA SET 8 AND PUTS THE
C SCRAMBLED VERSION IN A DATA SET 9.
C
1  INTEGER ITCOUNT (A-Z)
2  INTEGER CARD(2)
3  INTEGER SCR(12)
4  INTEGER A (70)
5  INTEGER IUCOUNT
6  INTEGER CLOCK1
7  REAL ITCLC
8  INTEGER J(1)2

C KUDGE FOR ATTACHING FNL 2
9  ICOUNT=0
10  CONTINUE
11  READ(I,900,END=2) CARD
12  ITCLC=CLOCK1+1
13  GOTO 1
14  END
15  WRITE(2,902) CARDS
16  GOTO 50
17  END
18  WRITE(6,950) ICOUNT
19  END
20  FORMAT(*'TIME USED WAS ',F8.4,' SECONDS')
21  END FILE 9
22  WRITE(6,950) ICOUNT
23  END
24  FORMAT(*'IT CLC',I17,*' CARDS WERE SCRAMBLED,'*)
25  WRITE(6,950) ICOUNT
26  END
27  FORMAT(*'END OF SCRAMBLED FILE FOUND, ',I8,*' CARDS READ')
28  RETURN
29  END
 tbsp. L.linfaXF.

VOL SER NOS= SYS02, US.

VOL SER NOS= CAMPASS.

SY1.UIT2.

VOL SER NOS= SYS02.

SY1.UIT3.

VOL SER NOS= SYSC3.

SY1.UIT4.

VOL SER NOS= SYS02.

SY1.UIT.

VOL SER NOS= SYS02.

SYSJUT.

VOL SER NOS=.

SY1.UIT5.

VOL SER NOS=.

VOL SER NOS= SYSC3.

AAAAAA,AAAAAAAA,AAAAAAAA,0000012 PASSED.

VOL SER NOS= SYSC3.

X.

VOL SER NOS= 2408.

DELETED.

VOL SER NOS= LIJ001.

DELETED.

VOL SER NOS= 2408, LIJ001.

DELETED.

VOL SER NOS= SYSC2.

DELETED.

VOL SER NOS=.

DELETED.

VOL SER NOS=.

VOL SER NOS=.

VOL SER NOS=.

HASP JOB STATISTICS -- 171 CARDS READ -- 312 I/O CALLS 434 SVC CALLS 590 LINES PRINTED -- 0.20 MINUTES CPU TIME.