DEVELOPING A FASTER IMAGE

Charles Sullivan
RunningMate Software
3001 I Street
Sacramento, California 95816

INTRODUCTION

This paper examines some performance concerns most of us have about TurboIMAGE. I have tried to examine topics which have not been empirically studied, such as the degradation caused by logging and the correlation between capacities and DBPUT performance for master datasets.

ABOUT THE BENCHMARKS

Any benchmarks shown here were run several times to ensure that the results presented are reasonable. Unless otherwise noted, the Series 48 and 70 had MPE disc caching turned on. The Series 48 had 3 megabytes of main memory, the Series 70 had eight, and the Series 950 had thirty-two. Each system had two or three disc drives which were a combination of 7933H and 7937H drives. The Series 48 and 70 were using either UB-Delta-1 or V-Delta-1 of MPE V and the Series 950 was running the 1.0 release of MPE XL.

TRANSACTION LOGGING PERFORMANCE

Transaction logging has not been treated with a great deal of rigor when discussing performance. For many years the received wisdom was that transaction logging exacted a severe performance penalty. Now the party line has become that the overhead associated with logging every DBUPDATE, DBPUT, and DBDELETE is negligible.

The database used to obtain the following results was configured as follows. The master dataset had a 10-byte key with a capacity of 9999, an entry size of 110 words and a blocking factor of four. The detail dataset had an entry size of 110 words, a blocking factor of four, and one search path. In all tests, 5000 records were added, updated, or deleted. Auto defer was off.

TABLE 1: DEGRADATION CAUSED BY TURBOIMAGE LOGGING

SERIES 70

MPE disc caching parameters Sequential fetch=96 Random fetch=32 Block on write=YES	Degradation	
	CPU Time	Wall Time
Master dataset DBPUT	18%	11%
Master dataset DBDELETE	20%	15%
Master dataset DBUPDATE	21%	19%
Detail dataset DBPUT	12%	5%

SERIES 48

MPE disc caching parameters Sequential fetch=96 Random fetch=16 Block on write=NO	Degradation	
	CPU Time	Wall Time
Master dataset DBPUT	18%	16%
Master dataset DBUPDATE	20%	15%

SERIES 950

32 megabytes main memory	Degradation	
	CPU Time	Wall Time
Master dataset DBPUT	29-57%	36-59%
Master dataset DBDELETE	19-45%	34-48%
Master dataset DBUPDATE	28-50%	31-60%

The results on the Series 70 and Series 48 were easy to reproduce. But tests run on the Series 950 had a wide range of values. Logging degrades performance much more on the Series 950 than on machines running MPE V.

Before you begin TurboIMAGE logging, the performance penalty of ten to twenty percent for most DBPUTs, DBDELETEs, and DBUPDATEs is worth pondering.

SELECTING A MASTER DATASET CAPACITY

B. David Cathell presented at the Interex conference in Los Angeles in 1984 a pioneering paper about master dataset capacities called *IMAGE: An Empirical Study*. He concluded that capacities for master datasets based on a prime number were not better (or worse) than non-prime capacities. He examined the synonym distribution over different capacities of the same data.

In the course of solving a severe response-time problem that we experienced, I had occasion to write a program which has allowed me to confirm Cathell's result and to extend it. The program simulates taking the current entries in a master dataset and loading them into a dataset of different capacity. The program determines the number of synonyms that will exist and a "clustering" index. This clustering index is the number of 50-record chunks which contain 50 entries. For example, a dataset which is empty except for entries in the first 100 records would have a cluster index value of fifty-one. The higher this value, the longer a DBPUT will take to execute, on average. [See *Identifying Opportunities for Performance Improvement* by George B. Scott in the 1986 Detroit Interex proceedings for data about clustering.]

On the next page you will find a sample of the values that were obtained for a dataset with a 20-character (X20) key and 29967 entries.

Cathell concluded that the only capacity to absolutely avoid was one which is a power of two. Look at the result for a capacity of 65,536. Here the cluster index becomes the "Custer" index--whoever chooses that capacity is going to get massacred by unhappy data-entry employees.

Given a particular set of data and a particular dataset capacity, you cannot predict its DBPUT performance without examining the distribution of ALL the data in the dataset. Here are some guidelines I now use:

1. Half-empty datasets usually produce an excellent cluster index. Datasets which are 60% full are acceptable, but 70% produces DBPUT performance which is erratic. Eighty percent and above is to be avoided. Selecting a prime number guarantees nothing.

2. Be very careful selecting capacities which are very close to being a power of two. From the following table, you can see that 65536 and 65537 are execrably bad, but 65540 would be tolerable.

TABLE 2: SYNONYM COUNT AND CLUSTER INDEX VALUES FOR SELECTED MASTER DATASET CAPACITIES

KEY TYPE = X20 DATASET ENTRIES = 29,967

New Capacity		Percent Full	Synonyms	Cluster Index
40,007		74.9	8,977	883
40,008		74.9	9,847	1,129
40,009	PRIME	74.9	8,927	1,314
40,010		74.9	9,160	1,183
.				
45,006		66.6	8,356	348
45,007	PRIME	66.6	8,268	225
45,008		66.6	9,608	239
.				
49,998		59.9	7,825	54
49,999	PRIME	59.9	7,697	15
50,000		59.9	8,737	17
50,001		59.9	7,593	44
50,002		59.9	7,704	0
.				
55,000		54.5	7,939	15
55,001	PRIME	54.5	7,130	8
55,002		54.5	7,281	0
.				
59,998		49.9	6,803	0
59,999	PRIME	49.9	6,613	0
60,000		49.9	8,262	0
.				
64,000		46.8	10,872	162
64,001		46.8	6,631	3
64,002		46.8	6,804	0
.				
65,535		45.7	10,092	1,772
65,536		45.7	16,165	14,165
65,537	PRIME	45.7	11,366	11,751
65,538		45.7	8,479	986
65,539	PRIME	45.7	7,372	245
65,540		45.7	7,501	161
.				
69,999		42.8	5,789	0
70,000		42.8	6,897	0
70,001	PRIME	42.8	5,802	0

SELECTING A GOOD VALUE FOR BUFFSPECS

TurboIMAGE on MPE V machines allocates an entire extra data segment for data buffering. I tried to determine if altering the number of buffers affected performance. My working assumption was that, in a multi-user environment, setting BUFFSPECS=16(1/120) would produce poor throughput compared to setting BUFFSPECS=64(1/120).

My results showed no difference between such divergent values for the BUFFSPECS parameter. I do not, however, consider this result conclusive. More testing needs to be done.

SINGLE THREADING

TurboIMAGE allows some multi-threading of database intrinsics. The Hewlett-Packard reference manual states that a "two level resource priority locking scheme is used within the DBB to allow single-buffer operations to access the control block concurrently. This involves DBGET, DBFIND and DBUPDATE processes. DBPUT and DBDELETE operations are unable to access the DBB concurrently. These multi-buffer operations must hold a global lock on the DBB throughout the operation." Although the wording is not exactly clear, it does appear that serial read DBGETs also lock the DBB until intrinsic completion.

Single-threading should only be a concern when most of your programs access a single database and your CPU is spending a good portion of its time paused for disc I/O. Although each CPU essentially holds all data in a single database, we have found that our CPUs are not often paused for disc I/O despite single-threaded DBPUTs. The reasons for this are three-fold: 1) MPE disc caching tends to make a system CPU-bound rather than disc-bound; 2) Handling terminal I/O for sixty users over an X.25 network consumes any free time the CPU might have; 3) All our serial reads bypass the DBB entirely (see the next paragraph), thus we have essentially made TurboIMAGE multi-threaded.

If you have performance problems because of single-threading, I recommend that you invest in one of the several software products available which executes fast serial reads and use it with your reporting programs and job streams. These MR-nobuff products do not lock the DBB and hence you should experience more database concurrency and throughput.

SPEEDING UP SERIAL AND CHAINED READS

Several software products are available which can speed serial access to datasets by a factor of three to ten. An early discussion regarding such MR-nobuff techniques can be found in *Overview of Optimizing (On-Line and Batch)* by Robert M. Green from the Interex 1982 proceedings in San Antonio.

Speeding up non-serial access to TurboIMAGE databases is less of a sure-thing. In developing I/O-Mate, it was found that the best overall performance for a calculated (mode=7) DBGET was achieved when just one block of data was fetched at a time. On the other hand, it was found useful to vary the fetch size for chained (modes=5,6) and random (mode=4) DBGETs. I/O-Mate dynamically adjusts fetch size based on the efficiency of recent fetches. I/O-Mate also tries to do things only once. For example, it keeps the two most recent lists for each dataset handy. I/O-Mate also caches the two most recently accessed datasets for even faster access. Finally, although it stores data in extra data segments, it never performs an EXCHANGEDB procedure call. All these optimizations allow I/O-Mate to outperform a "vanilla" DBGET(modes=4,5,6) by fifteen to fifty percent.

APPENDIX: HASHING ALGORITHM

```
DOUBLE PROCEDURE HASH(KEY,HASHEDKEY,KEYLENGTH,CAPACITY);
    VALUE HASHEDKEY,KEYLENGTH,CAPACITY;
    ARRAY KEY; LOGICAL HASHEDKEY; INTEGER KEYLENGTH; DOUBLE CAPACITY;

  BEGIN

    PUSH(STATUS); ASSEMBLE(TRBC 2); SET(STATUS);

    TOS := CAPACITY;
    IF HASHEDKEY THEN
      BEGIN
        ASSEMBLE (LDD KEY, I;  LOAD KEYLENGTH;  DUP, STAX;
                  BRE HASH1;  DLSR 16;
        HASH1:    DLSL 1;  LOAD KEY;  ADXA, LDXA;  INCA, NOP;
                  LSR 1;  DXCH, NOP;
        HASH2:    DECM S-2;  BLE HASH3;  DECM S-3;  DECM S-3;
                  LDD S-3, I;  DDUP, NOP;  LDI 31;  LDIV, DELB;
                  STAX, DXCH;  DCSL 1,X;  DADD, NOP;  BR HASH2;
        HASH3:    DXCH, DDEL;  DLSR 1);
      END
    ELSE
      BEGIN
        ASSEMBLE (LOAD KEY;  LOAD KEYLENGTH;  LADD, DECA;  DECA;
                  LDD S-0,I);
        IF KEYLENGTH = 1 THEN ASSEMBLE (ZROB)
        ELSE ASSEMBLE (DLSL 1; DLSR 1);
        TOS := TOS - 1D;
        IF < THEN
          BEGIN
            ASSEMBLE (DDEL,DEL);
            GOTO ALL'DONE;
          END;
        ASSEMBLE (CAB,DEL);
      END;

    ASSEMBLE (LOAD S-3; STAX);
    IF <> THEN
      ASSEMBLE (LDXA;  SCAN 0;  XAX, NEG;  STAX, DDUP;  DLSR 16,X;
                DELB, LDIV;  DEL, DUP;  LDD S-5;  CAB, LMPY;
                DXCH, MPY;  ZERO, DADD;  DSUB, DZRO;  INCA, DADD;
                LDD S-3;  DSUB;
      CHECKCC:  BE EQ;  BG MORE;  LDD S-3;  DADD;  BR CHECKCC;
      MORE:     DDUP;  LDD S-5;  DSUB;  BL OK;  DXCH, DDEL;  BR MORE;
      OK:       DDEL;  DXCH, DDEL)
    ELSE ASSEMBLE (DXCH,DELB;
                   LDIV, ZROB);

    ASSEMBLE (DZRO, INCA;
    EQ:       DADD);

ALL'DONE:
    HASH := TOS;

  END;  << DOUBLE PROCEDURE HASH >>
```