SETL Newsletter 99 and BALM Bulletin 21 S. Brown

PAGING, THE QUICK AND DIRTY WAY Jan. 31, 1973

A version of BALMSETL with paging is available. The implementation is loosely based upon the description in SETL Newsletter 86. It proved to be only slightly more difficult to implement than anticipated. The SETLB user should not expect the paging version to solve all of his space problems as this version represents, at best, an interim solution. The user will find himself once again confronted by the dilemma of trading core size against running time.

There follows (1) a brief description of the implementation, (2) some results of test runs, (3) conclusion and suggest improvements and, if the user is still with us, (4) the necessary job control to run the paging version.

Implementation

A series of new BALM primitives (described in BALM Bulletin 19 were implemented. These extended the BALM capabilities to include opening random files, reading and writing records randomly, and implicitly passing arguments to a procedure. Next several BALM procedures were written which control paging in the following fashion.

MAKEPAGE -

1. scans the symbol table and makes a list of procedures which may be paged to disk.

MAKEPAGE may be called as often as desired. Each call makes procedures added since the last call available for paging.

SELECT -

1. Groups the procedures on the list supplied by MAKEPAGE into pages.

Instead of each procedure being a page several procedures make up one page and are paged together.

SWAP -

1. Writes the pages to disk.
2. Replaces the paged procedure with a small procedure which either executes the procedure itself or calls PAGINREC to retrieve it from disk and executes it.

PAGINREC -

Is called whenever a procedure must be paged in from disk.

1. Checks to see if

SIZE(PAGE(I)) + CURRENT-CODE SIZE > MAXCODE

If not the PAGE(I) is brought in.

If yes the latest used pages in core are released and then PAGE(I) is brought in.

2. Updates CURRENT-CODE-SIZE appropriately.

SOME RESULTS OF TEST RUNS

	NO PAGING			PAGING			
PROGRAM	TIME (sec)	Garbage Collec.	Field Length	TIME	Garbage Collections	Field Length	MAXCODE SIZE
BALMSETL	119	24	170K	154	34	170K	8,000
MEDIAN	13.9	2	177K	18.9	4	177K	8,000
MEDIAN	14.7	4	170K	17.9	2	170K	10,000
MEDIAN	insuff. space	3	160K	19.3	4	160K	10,000
MEDIAN	-	-	-	19.9	5	150K	10,000
MEDIAN	-	-	-	24.1	9	140K	10,000
MEDIAN	-	-	-	insuff. space	3	130K	10,000
TREEPRINT	26.3	7	177K	50.1	16	177K	12,000
TREEPRINT	-	-	-	42.5	13	177K	14,000
TREEPRINT	-	-	-	56.4	27	160K	14,000
PERM	15.7	3	177K	23.0	4	160K	12,000

CONCLUSIONS AND SUGGESTIONS

Any paging scheme which acts independently of the garbage collector is bound to be far from optimum. First of all the amount of code to allow in core represents another parameter to adjust. If it is too small to accommodate the procedures necessary for running the program a great deal of paging takes place. This, in turn, increases the number of garbage collections as the garbage

collector must be invoked to reclaim space used by code blocks which have been paged out. The importance of an appropriate setting of MAXCODE is illustrated by the difference between runs 1 and 2 of MEDIAN (with paging) in which the program ran faster with fewer garbage collections even though the field length was decreased.

The setting of MAXCODE and field length are clearly somewhat dependent upon the program. However, the relationship between the two can be computed.

In BALMSETL without paging about 30,000 words are devoted to code. Paging reduces this to 6000+MAXCOD. Therefore if MAXCODE is set to 12,000 (which is the default) then the total amount of space devoted to code is 18,000 words. The field length can then be reduced by 12,000 words or about 30K. This will, of course, increase the number of garbage collections and the total running time.

The current implementation could be improved by grouping procedures, which should occur in the same page, intelligently. This would cut down the amount of paging and is probably the easiest step to increase efficiency. Further improvement could be gotten by coding paging at a lower level. Rather than a BALM procedure to retrieve code from disk the CALL primitive could be modified to do the work. Clearly paging when the garbage collector runs out of space would be an improvement but that suggests that the paging scheme described in BALM Bulletin 17 should be implemented instead of continuing work on the present version.

HOW TO USE THE PAGING VERSION

```
JOB CARD
RFL, 66000.
ATTACH(SETLB,SETLB)
SETLB.
ATTACH(BALM4,PBALM4)
ATTACH(BLM4SVD,SAVESETLP)
RFL,NNNNN.
BALM4(SETLOUT)
E-O-R
 SETLB PROGRAM
E-O-F
```

A user may adjust the amount of core reserved for code by including including the following card in his SETLB program.

```
DO; SETMAX(N); COMPUTE;
```

where N is the desired size. A user may page his procedures by executing DO; MAKEPAGE; COMPUTE;

All procedures compiled before the call to MAKEPAGE will be paged to disk and brought in as required.