The role of CAFS at OUCS : Working Paper for the 2900 Replacement Committee

L. Burnard

1. What CAFS is.

1 Central processor unit (CPU) time is used up typically in one of two ways: either in intensive processing (e.g. in mathematical work) or in data manipulation (e.g. in loading data into store in order to determine whether or not it is to be processed). CAFS provides an alternative to the usage of CPU in the latter task. The task of selecting particular records from a large dataset can be entirely delegated to CAFS; CPU resources are used only to process records once they have been selected. The greatest benefits in using CAFS will therefore be found when the "hit-rate" (the proportion of records to be processed relative to the number of records searched) is low. With a 100% hit rate there will be no advantage in using CAFS at all; with hit rates less than 30 or 40 percent the amount of CPU time used may be a quarter or less when CAFS is employed. (See further Appendix 2)

2 Manipulation of textual data is also a highly CPU-intensive activity. Text has to be "tokenised" before it can be searched, if access by particular word or word-pattern is required. Efficient access using such tokens as key will normally require the construction and maintenance of index files. If many categories of index are required, the size of such indexes may exceed that of the original

data file. With CAFS, no indexes are necessary, although the text still has to be converted to a searchable format (known as Self Identifying Format or SIF) in which tokens are tagged with their name and category.

3 If a job requires very few CPU resources it becomes feasible to run it online at a MAC terminal; in some circumstances, CAFS can thus completely change the way in which users interact with their data.

4 Because many of the problems addressed by CAFS have been traditionally addressed by database management systems (DBMS) or comparable software, there is a natural tendency to view CAFS as a “database machine” of some kind. This is only partially true: some database operations (notably anything to do with updating) are not assisted in any way by CAFS, while some functions of CAFS (notably its ability to search text rapidly) are not usually supported by traditional DBMS.

1.1 Hardware

5 On ICL 3900 machines CAFS is now a standard component of the disc controller subsystem. It comprises four distinct processors which are programmable by the mainframe CPU and which collectively have the effect of filtering the data flow read from the discs before it is processed by the CPU. Appendix 1 gives a technical description of the way the hardware operates.

1.2 Software

6 ICL offer a range of software capable of exploiting CAFS. At one extreme, as might be expected, ICL’s own strategic product line (IDMS, QM, RCI ...) all use CAFS intensively. At the other extreme, an extension to VME known as the CAFS Search Option (CSO) can make almost any software trigger off a CAFS search, even after it has been compiled. In between, there is a library of routines, the Direct CAFS Interface (DCI), which offers programmers the ability to exploit all features of the hardware from their own purpose-written software.

1.2.1 Querymaster

7 Querymaster (QM) is a “user-friendly” relational-style query processor. We have used it extensively as a way of providing a simple and versatile means of interrogating IDMS and PDS databases; it also has some simple text searching capabilities and can process most forms of SIF records.

8 QM invokes CAFS invisibly, at its own discretion as it were, whenever to do so would significantly improve its performance. Typically it is used when searching through IDMS databases or indexed files for non-keyed data-items, when it can reduce CPU-time usage by one or even two orders of magnitude. This is particularly important when files are to be “joined” on non-keyed items.

9 Its chief drawbacks in our environment are (a) its comparative inflexibility - QM commands cannot be embedded in a high level programming language and its output facilities are limited, though flexible (b) its lack of support for all CAFS facilities, particularly as regards text processing and (c) its dependence on Data Dictionary System (DDS). QM uses a special module, known as a query view, which has to be created from data entered in a data dictionary. This is entirely appropriate in any commercial or other large scale data processing situation, but represents a significant overhead for us, both in service support (DDS is definitely not a user-friendly product) and in filestore usage. It should be noted however that the marginal cost of creating a query view for an existing IDMS or other system already defined in DDS is trivial.

1.2.2 CAFS Search Option

10 The CAFS Search Option (CSO) is the easiest software interface to CAFS. It enables CAFS searches to be invoked by existing Fortran or Algol68 programs and packages, by means of an SCL “hook.” Files to be CAFS searched using CSO must have fixed length, fixed-format, fields containing only character or fixed-point binary data.

11 The user specifies the record-structure and the search criteria by means of the VME command SET_CAFS_CRITERIA (STCC) which is executed after the file to be searched has been assigned and before the program is executed. The first READ executed by the user program initiates a CAFS search; all subsequent READs will return to the user program only records satisfying the search condition set by the call of STCC. The user program thus “sees” only hit records and any selection carried out within it is additional to that specified by STCC.

12 CSO is a very simple and attractive way of performing simple CAFS searches at little or no development cost. An existing program can run unmodified except for the call to STCC at three or four times the speed, (depending on the hit rate and what is done with the retrieved records). CSO can also enhance dramatically the performance of some favourite software packages (e.g. BMDP, SPSS) which are otherwise somewhat expensive to run on large data sets. Although it only offers a subset of CAFS facilities, it is a useful subset matching many user requirements. It should be noted

however that the benefits gained by using CSO are inversely proportional to the proportion of the data set being processed: i.e. with a 100% hit rate, there is no benefit to be gained by using it all. (see figures in appendix 2)

1.2.3 Direct CAFS Interface (DCI)

13 This is a library of routines which can be called from any high level language (Cobol, Algol68, S3, Fortran, SCL...) and which together give the programmer access to all facilities of the CAFS engine from within an application program. It is the programmer's interface; as such it is no more difficult to use than say a library of graphical or numerical routines.

14 Amongst the facilities offered are:-

- * Access to several files under program control
- * Access to a wide range of SIF handling routines
- * Ability to pre-define conditions
- * Access to some arithmetic functions e.g. counts
- * Ability to perform fuzzy and masked matching

Of particular importance in our environment is the fact that DCI calls can be simply embedded in existing programs or built into new ones. CAFS facilities can thus be tailored to user-requirements and the learning curve required for competent programmers is very shallow.

1.3 Usability

- (a) CAFS is unique to ICL equipment. No other manufacturer has any device which supports the full range of its functions, or even more than one of them. Neither of the devices we have investigated which come closest to it (the Britton-Lee Intelligent Database Machine and Gould's Memex) provides a complete alternative to the current system.

- (b) CAFS should not be regarded as a database accelerator. It can improve performance of some database operations, but not all. It does to some extent simplify the database design process for IDMS databases, because fewer indexes etc. are required. For some applications, CAFS may provide an alternative solution to problems which have in the past necessitated a database solution, for example those involving large unstructured text files, or even just filtering large sets of structured records (e.g. hospital data).
- (c) CAFS requires very little special-purpose programming to achieve large performance enhancements. At the simplest level, two additional lines of SCL are needed to cause a standard package such as BMD-P to initiate a CAFS-assisted search through a data file. Such data files do not have to be transformed or moved to a separate device to be CAFS-searchable. At the most complex level, where textual data is to be searched, the text file has to go through a one-off conversion process, the complexity of which will be determined by factors such as whether or not a standard alphabetic encoding has been used for the machine readable text.
- (d) The Direct CAFS Interface offers access to the facilities of the CAFS system at a level which is appropriate to our environment. Any competent programmer can construct his own CAFS-based software. The level of service support for this type of facility is greatly reduced from that needed e.g. for database systems.

1.4 Future Developments

15 CAFS is not perfect, nor is it a panacea. Indications from ICL are that some of its more obvious short-comings are being addressed by developments already underway. It is already clear that the next generation of CAFS will be engineered in VLSI giving considerable increase in functionality as well as in performance. Amongst the functional improvements widely tipped are implementation of more relational functions (notably the join) and expansion of text processing primitives, e.g. to support context-sensitive searching and left hand truncation of search terms. It is probably worth noting also that CAFS is now clearly regarded as central to the company's future development, in a way which was not so clearly true a few years ago.

2. Current and future CAFS-based systems

16 This section updates the description of our usage of the system given as an appendix to the 1985 Annual Report (AR).

2.1 IDMS and PDS database systems

17 The process of converting PDS to QM based systems has been to some extent automated. We now have some dozen operational QM/CAFS systems which started off as PDS systems. Many recent applications are simple one-user systems which might perhaps better suit a microcomputer, but there seems to a steady demand for database facilities from users without access to micros. Perhaps fortunately in view of the current uncertainties, no major new database applications have appeared additional to those described in AR.

18 CAFS is clearly a vital component in any database systems developed using our current range of database software; in its absence QM can be very inefficent, while its presence greatly simplifies the task of database design for IDMS applications. One advantage of QM over most such products is that it does permit integration of text files with database systems: one application (the Protestant Cemetery database) is already experimenting with this facility, and it seems probable that similar historical applications will follow.

2.2 Text Processing systems

19 This has been an area of major activity. A text-searching package written in Algol68, originally simply to test the functions of DCI, is now being developed for use with a number of large data sets. These include the Shakespeare database mentioned in AR, and three distinct library catalogues (the Bodleian pre-1920 catalogue, the Inter Collegiate Catalogue of Early Printed Books and the Radcliffe Science Library's Serials File). An article in the November issue of the ICL Technical Journal describes these projects in some detail.

20 Enhancement of this software will continue, with a view to making it more user-friendly and to extending the range of text-specific operations supported directly by it. It is planned to create a large database of Greek text in CAFS searchable format, following a number of user requests.

2.3 Data selection systems

21 CSO only became available to non-COBOL users in November 1985; our early tests indicate that it will be of considerable importance to a section of the user community which has received less benefit from CAFS than might have been expected. Many users, particularly in the animal and human sciences, have very large data sets, often held on magnetic tape, from which subsets are routinely extracted for further analysis by standard or custom-written statistical packages. Conversion of such large data sets to true databases as a way of improving the overall efficiency of such operations is not always an appropriate solution. Use of CSO or DCI would greatly increase the overall efficiency of such activities, very simply.

22 A user guide describing the use of CSO in some detail is in press, as is an article in the Newsletter drawing attention to the new CAFS facilities. It is also planned to give two seminars on using CAFS later in Hilary term.

3. Alternatives to CAFS

23 Fundamentally, there is no alternative to CAFS not involving additional machine resources or software effort. If no ICL system were available at OUCS, it would therefore be necessary to investigate ways of relocating the work load currently and potentially supported by it.

3.1 Database Systems

24 Clearly, the extent to which the absence of CAFS would impact future database systems very much depends on the efficiency and nature of the future DBMS. If this were based around the Britton-Lee IDM, for example, straightforward database style applications would be at least as efficient, and probably (given the IDM’s hardware support for a greater range of relational operations) better. If a more traditional software based dbms were chosen, it is evident that those database operations currently supported by CAFS would need to be supported by the DBMS. This might to some extent increase the design and support effort needed, although such increases might well be offset by an overall reduction in the level of support needed for DBMS more up to date than our current system.

25 Many of our existing database applications are to some extent involved with text processing which neither the IDM nor relational DBMS such as INGRES or ORACLE supports particularly well. Support of such systems would require more resources than at present, both in terms of machine time and filestore usage.

3.2 Text based systems

26 CAFS greatest strength is its support for a wide range of text processing activities, which are also coincidentally among the facilities most in demand at OUCS. The only hardware-based alternative to CAFS in this area is Gould's revamped MEMEX machine, which is inferior in a number of respects to CAFS: firstly it requires that the text data be converted to an entirely different "tokenised" form, and secondly it does not support categorisation of tokens within the search unit. Text processing facilities can of course be provided by software packages such as STATUS, but far less simply and at a much greater cost in machine resources. It should also be noted that CAFS offers a range of software tools rather than a finished package: this means that the service can continue to develop products appropriate to its users' often somewhat esoteric requirements, which are rarely supported directly by off-the-shelf retrieval packages.

3.3 Data selection systems

27 Again, it is hard to see what alternatives exist As a way of enhancing the performance of the simple "pick out all the records with a star in column 12" type application, the only alternative to CAFS is simply to enhance the processor power, perhaps extending it with such devices as extensive cache memory. It is debatable whether this is a valid use for scarce computing resources.

4. Conclusions

- * CAFS enhances and simplifies some, but not all, database operations
- * It is of particular importance in text searching applications
- * There will always be a threshold beyond which some activities become too expensive for the average user; CAFS moves that threshold considerably further than the limits of available CPU power.

5. Appendix 1 : CAFS hardware functions

28 The following diagram shows how the four parts of the CAFS unit interact.

29 The four units act as follows:-

- This identifies logical records and subdivisions of records within the byte stream read from the disc. Records may be of fixed or variable length; unless they are in SIF, only one variable length field is permitted and it must be the last part of a record. The record-structure is defined by the user at run time: the firmware in the LFU being passed the information necessary to split up the byte stream when the first CAFS search is initiated on a particular file.
- This converts the input byte stream into a stream of records as they are to be passed back to the CPU (Some fields may be ommitted). This process is carried out simultaneously with that carried out by the Search Evaluation Unit
- This comprises sixteen key-channels (KCs) and one Search Evaluation Processor (SEP). The input byte stream is multiplexed across the sixteen key-channels, each of which is loaded with data sufficient to perform a single test (e.g. “are bytes 1 to 4 LE 0000 ?”), and an optional mask. The data needed for the test is loaded in by the user at run time, in the same way as the record structure is passed into the LFU. Each KC then signals the success or failure of the condition it has evaluated to the SEP which can then determine whether the record as a whole is to be retrieved or not, using either normal Boolean or quorum logic.
- This contains a 24 Kb buffer store in which hit records can be accumulated for some simple arithmetic functions to be performed on them, e.g. totalling, counting, averaging and calculation of maxima and minima; it is also possible at this stage to compare the values of two fields within the same record as a further retrieval criterion.

30 The crucial point to note about this architecture is that the overall data through-put is limited primarily by the speed at which data can be read from the discs (rather more than 2 Mb/second on 3900 series discs; about one Mb/second on 2900 series) No component of the CAFS system interrupts the flow of data from the disc by buffering or storing it, other than the RP which normally is dealing with only a small fraction of the total number of records processed by the unit.

6. Appendix 2 : Some performance statistics

31 These figures were obtained during the normal user service, using CSO to perform CAFS searches.

6.1 (a) Simple Fortran program

32 The program reads a file of 20,000 80-byte records and tests the value of one column to determine whether or not the record is a hit. A count of hit records is maintained.

	cpu usage	
hit rate	**without CAFS**	**with CAFS**
%	seconds	
0	8.941	2.639
10	9.005	3.330
30	9.101	4.568
60	9.395	6.853
100	9.152	10.241

6.2 (b) Simple package usage

33 SPSS job producing a simple Condescriptive set of statistics for the same data set, selecting on the value of one variable

	cpu usage (secs)	
hit rate (%)	**without CAFS**	**with CAFS**
10	95.141	17.495
30	90.711	29.809

60	97.744	65.223

34 BMD-P job selecting on the value of one field from a file of 69,000 twenty-byte records and printing the hit records.

0.005	232	3.1

6.3 (c) \2Algol68/DCI text searching\1

35 These figures show the amount of OCP used to perform some typical text processing tasks against (1) the Shakespeare Database (about 16 Mb of free text) and (2) a sample from the Bodleian pre-1920 catalogue (about 24 Mb). The tasks are as follows:

- (1) Count the records but not retrieve any
- (2) Retrieve all records containing a specified word
- (3) Give separate counts for records containing one of a list of six different words, and retrieve any records containing at least three of the specified words

File	Records Searched	Task	Records Retrieved	CPU used
1	162313	1	0	0.263
		2	4383	22.484
		3	25	0.927
2	83607	1	0	0.726
		2	489	14.942
		3	13	1.845

36 No comparable figures could be obtained for non-CAFS usage as the software used does not operate without CAFS.

6.4 (d) Using Querymaster

37 Here, Querymaster is used to search a large PDS database, testing the values of the specified numbers of non-key fields. In the third example below, two tables are joined on a non-identifier column.

Conditions	Records Searched	Records Retrieved	CPU usage (seconds)	
			with CAFS	without
4	1773	14	4	7
2	69185	1245	9	322
2 (+join)	69185*1044	195	19	>650

BIBLIOGRAPHY

NOTES

ABSTRACT

CAFS (Content Addressable File Store) is one of the features of the current ICL 2988 system which is unquestionably unique, both intrinsically and in the contribution it makes to the facilities available at OUCS. This paper addresses three subjects which the Replacement Committee might care to take into account when determing any contribution that ICL equipment, notably CAFS, might make to the new configuration. The topics addressed are (1) what exactly CAFS is (2) current and short-term plans for its use and (3) possible alternative strategies in its absence.

AUTHORS

L. BURNARD