An Empirical Performance Study of the Ingres Search Accelerator for a Large Property Management Database System

Sarabjot S. Anand
Dept. of Information System
University of Ulster
Northern Ireland
ssanand@uk.ac.ulster.ujvax

David A. Bell
Dept. of Information Systems
University of Ulster
Northern Ireland
dbell@uk.ac.ulster.ujvax

John G. Hughes
Faculty of Informatics
University of Ulster
Northern Ireland
jgh@uk.ac.ulster.ujvax

Abstract

The property management database system under development at the Northern Ireland Housing Executive (NIHE) is a large relational database system. The application system has a high expected transaction processing rate - approximately 37000 transactions per day (most of them accessing mutliple tables) from about 250 on-line users. Performance is of critical importance in its success. In this paper we consider the effect of the Ingres Search Accelerator on the transaction processing efficiency of the system. The performance enhancement brought about by SCAFS (ICL's current version of the well-known Content Addressable File Store, CAFS [BABB79, BABB85,COUL72] - the heart of the Ingres Search Accelerator) for different file organisations is assessed. Recommendations on how the performance of SCAFS can be improved by tuning certain parameters is provided. We also provide a rough guideline as to when the Ingres Query Optimizer "decides" to use SCAFS for different file organisations and point out deficiencies in this decision making process. We conclude this paper by recommending techniques that may be employed to increase the role of the Ingres Search Accelerator in Ingres database systems.

Proceedings of the 20th VLDB Conference
Santiago, Chile, 1994

1. Introduction

The Property Rent Accounting and Waiting List (PRAWL) database system under development at the Northern Ireland Housing Executive (NIHE) is large - NIHE is the largest "landlord" in Europe. We refer to it in the rest of the paper as the NIHE database system. The system is being implemented using Ingres/Star on 11 ICL's DRS6000 machines. The approximate size of the PRAWL database is 20 GB spread over 5 NIHE regions. The expected work load on the system from about 250 on-line users is 37000 multi-table transactions per day.

A database system of this size and transaction rate clearly requires that considerable attention be given to performance. One 'performance enhancer' available in the applications environment is the Ingres Search Accelerator released by ICL in partnership with Ingres Corp. in 1991. As the name suggests, the Ingres Search Accelerator (ISA) is claimed by its designers to accelerate searches on Ingres tables thus enhancing the overall performance of the database system. In this paper we investigate whether the ISA lives up to its name and try to gauge its usefulness to the PRAWL database system.

The chief potential benefit of using SCAFS is clearly to enhance performance of a database application. Now performance is usually assessed in terms of response time, throughput or utilization of system resources. Simulation, analytic or measurement studies can be used to evaluate these indices. The present paper reports on a performance measurement study of SCAFS mainly using the responsiveness and, to a lesser extent, utilization indices.

Leung et. al. [LEUN85] studied the efficiency of the CAFS 800 but that analytic study is significantly different from ours for two basic reasons :

- SCAFS is very different from the CAFS 800 and so many of the objectives and parameters of the study carried out by Leung et. al. are no longer valid e.g. degree of amplification of the drive and cell size, which were characteristics of the EDS60 discs but are no longer meaningful. The parameters that effect the performance of SCAFS are more at the design level of the database e.g. file organizations and indexes. The performance of SCAFS in a multi - user environment is affected by scheduling on the SCAFS board and the number of locations at which the table being searched is located. These are the parameters that we take into consideration keeping in mind the fact that the NIHE database system is a large multi - user system with a high transaction processing rate.
- Our tests were performed on real data from the property management system database of the Northern Ireland Housing Executive, using measurement rather than an analytic model, as the previous study did.

Our objective in the present study is two fold. First we are looking for reasons why the performance of the newly-installed NIHE database system was lower than expected, despite the availability of SCAFS. There is a lack of practical guidance in the literature for development of large relational systems and we hope that by sharing our experience in this paper other users may benefit.

Our second objective is still pragmatic, but it has a more scientific thrust - to gain insights into the applicability of a backend filter architecture (SCAFS here) for different design profiles and to study how other database system modules e.g. the query optimizer, could utilize this information.

We first quantify the precise improvement in performance brought about by SCAFS over direct access search devices for different file organisations. We study the effect of different file organizations and query result sizes, i.e. the number of 'hit' tuples, through a series of simple queries made on the NIHE database.

Secondly, during the investigation of SCAFS we found that certain SCAFS and INGRES parameters affect the SCAFS performance. So we report on their effects. In particular we assess the effect of the SCAFS scheduling parameter and the number of locations over which a table being searched is divided.

Thirdly, we give a rough guide to when the Ingres Query Optimizer decides to employ the Ingres Search Accelerator, and critically examine cases where the decision made by the Optimizer may not be the 'best' in practice.

The rest of the paper is arranged in the following format. Section 2 describes the NIHE application whose performance enhancement is our prime concern in this paper. In section 3 we discuss the improvement in performance brought about by using SCAFS for scanning tables which use different file organizations and also consider the effect of compressing data. This is followed by a discussion of the effect of the SCAFS Scheduling Parameter on the performance of SCAFS in section 4. Section 5 discusses the effect of data placement on the performance of SCAFS. A rough guide to how the Ingres Optimizer decides on using or not using SCAFS when executing a particular query is presented in section 6 and section 7 concludes this paper and discusses on-going and future work involving SCAFS that is being undertaken in the authors' laboratory.

2. The NIHE Application

The Northern Ireland Housing Executive, established in 1971 by the Housing Executive Act (Northern Ireland), is the largest body of its kind in western Europe. The organisation is geographically distributed around 70 locations spread throughout Northern Ireland.

In 1987, the NIHE carried out a review of the state of their computerisation. The vital role of computerisation in the Housing Executive was recognised and a decision was made to enhance the present system.

Computerisation in the NIHE had started in the 1970's with a number of batch processing systems, mainly for financial accounting, being developed. In the 1980's a distributed network connecting each of the district offices of the Housing Executive was developed for on-line queries to their "tenant management" systems. The survey carried out in 1987 highlighted the gap between the NIHE's then current management performance in terms of innovation in policy development and its information technology resources and a decision was made to improve its use of information technology.

The NIHE decided to develop a ten year systems strategy detailing the contribution of information technology to corporate aims and objectives. The scope of the review included identifying applications that were required by the NIHE, determining their purpose, ranking them in an order of priority and identifying the most appropriate hardware and software strategy needed to deliver them. The result was a decision to procure a new integrated computer based tenant management system encompassing a property database, rent accounting and waiting lists. The core component of this tenant management system is the large property management database. The expected size of the database is approx. 20 gigabytes and the expected number of transactions per day is 37000, executed by approximately 250 concurrent, on-line users.

Transaction complexity of an application is normally measured by the the number of tables accessed per

transaction. For the NIHE the transaction profile is as follows: 4% of the transactions are queries accessing 1-3 tables, 59% are queries accessing 4-6 tables, 1% are queries accessing 7 or more tables, 29% are updates accessing 1-3 tables, 6% are updates accessing 4-6 tables and 1% are updates accessing 7 or more tables. Clearly this is a more complex transaction profile than normal OLTP applications like banking and airline reservations that normally have a higher transaction rate but lower transaction complexity.

The following specific objectives were outlined for the new tenant management system, in addition to the usual consistency and security objectives [NIHE88]:

- To increase efficiency in the performance of Property, Rent Accounting and Waiting List tasks.
- To improve the quality and level of information to managers, staff and customers.
- To improve control
- To provide "value for money"
- Minimise batch updates

In collaboration with ICL, the NIHE are currently developing a distributed property management database system called PRAWL, which will provide the Housing Executive with the facilities outlined above. The system is being developed in INGRES/STAR and uses ICL's DRS6000 machines. A large relational database system like PRAWL poses enormous problems to the implementors who need to look to the database research community for assistance. In this paper we discuss our experiences with investigating the possible use of ICL's SCAFS database filter machines in enhancing the performance of the NIHE system. Phase 1 of the project is now complete and is on-line. Phase 2 of the application is now under development. While the tests reported in this paper are based on the phase 1, it has also provided useful insights into performance enhancement for phase 2.

3. SCAFS Vs DASD

In our experiments to quantify the improvement in performance over the conventional DASD (direct access storage devices) approach brought about by the use of SCAFS, we considered the response time (CPU time + I/O time) of a query and the CPU time required for the query's execution. Performance indices were evaluated by measurement in the property management database.

Queries used in the experiments were made on an existing table from the application with 113922 tuples. Table 1 shows the size of the table for the four different file organisation types tested . The table was stored on a 1.2 GB disk with a data transfer speed of 1.9 Mb per second. As SCAFS "can scan data as fast as the data can be read off the disk" [INGR91], the time taken by SCAFS to scan a table should be dependent only on the size of the table. Table 1 also shows the theoretical scan time (= file size in Mb /1.9) required by SCAFS to scan the tables. Note that the difference in file sizes is due to the different fill factors and number of overflow pages of the table for different file structures.

Table 1 : File Sizes and Theoretical Scan Time of Experimental Data

File Organisation Type	File Size (Mb)	Theoretical Scan Time (sec)
Heap	28.482	14.99
Hash	79.204	41.68
ISAM	88.338	46.49
BTree	36.574	19.249

Tables 2, 3 show the effect of SCAFS on response time and CPU time and Table 4 shows the effect of data compression on SCAFS, for queries with no 'hit' tuples and 60000 'hit' tuples respectively. The enteries in these tables show the ratios indicated in the top, left-hand corner. As can be seen from Table 2, for uncompressed data the advantage of using SCAFS decreases for the larger number of 'hit' tuples. This is understandable because as the number of 'hit' tuples increases the I/O channel traffic increases, reducing the advantage of using SCAFS - the main gain only being in the CPU idle time while the table is being searched by SCAFS. For compressed data we find that, for hash and BTree file organisations, the effectiveness of SCAFS increases as the number of 'hit' tuples increases.

Table 2 : Effect of SCAFS on Response Time

No SCAFS	Uncompressed		Compressed	
SCAFS	0	60000	0	60000
Heap	2.94	2	3.167	1.52
Hash	2.667	2.296	2	2.57
ISAM	3.877	3.344	2.67	2.2
BTree	3.42	2.14	2.33	2.54

From Table 3 we find that the savings in CPU time using SCAFS are enormous for queries with the small number of 'hit' tuples but for the larger number of 'hit' tuples, the saving reduces and is sometimes even less than 2 fold when the number of 'hit' tuples is approx. 60000. Once again this is understandable because with a larger number of 'hit' tuples, the CPU is sent more data from SCAFS, increasing its load.

From Table 4 we find that for hash and BTree file organizations the benefit of using compressed data in terms of response time increases for the larger number of 'hit' tuples. Also for hash file organisation, using

compressed data increases the CPU time required for queries with the larger number of 'hit' tuples. This highlights the fact that the advantage of using SCAFS varies with the file organization type and great care needs to be taken in deciding when to use SCAFS for different file organizations (see section 6).

Table 3 : Effect of SCAFS on CPU Time

No SCAFS / SCAFS	Uncompressed		Compressed	
	0	60000	0	60000
Heap	106.33	1.62	95.875	1.42
Hash	97.34	2.91	95.428	1.718
ISAM	127.4	3.29	85.7	1.51
BTree	97.34	2.02	125	2.54

Table 4 : Effect of Data Compression on SCAFS Performance

Uncompressed / Compressed	Response Time		CPU Time	
	0	60000	0	60000
Heap	2.834	1.2	1.5	1.106
Hash	1.5	1.928	1.857	0.94
ISAM	5.44	2.44	2.5	1.14
BTree	1.05	1.458	1.89	1.09

CPU Time Vs Output Size
Heap File Organization

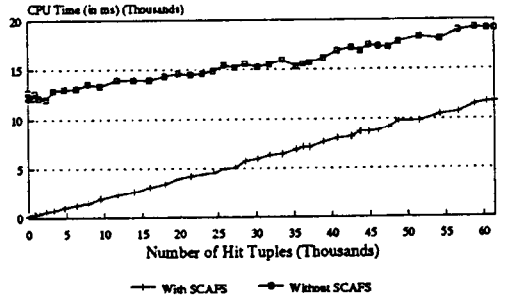

fig. 1(a)

CPU Time Vs Output Size
ISAM File Organization

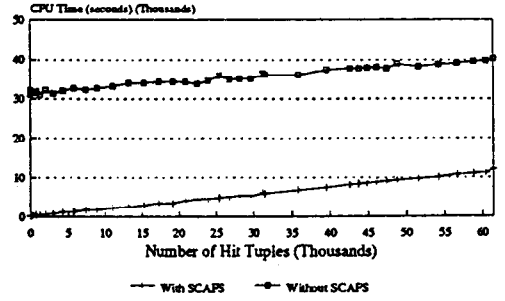

fig. 1(b)

Response Time Vs Output Size
Heap File Organization

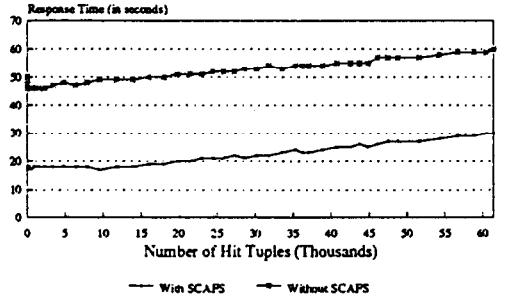

fig. 1(c)

Response Time Vs Output Size
ISAM File Organization

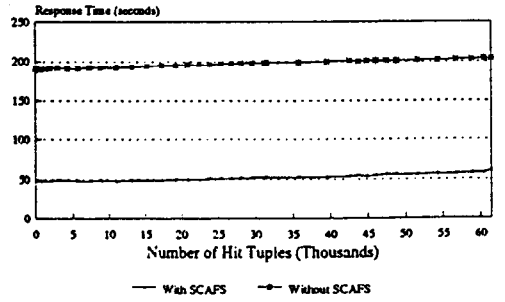

fig. 1(d)

The graphs in figures 1(a - d) show more comprehensively, how the response time and CPU time for queries varies as the number of 'hit' tuples vary from 0 to 60000 for heap and ISAM file organizations. The graphs in fig. 1(e - h) show the variation in response time and CPU time when using compressed data. We limit our discussion here to only heap and ISAM file organizations. For a detailed discussion including BTree and Hash file organizations we refer the interested reader to [ANAN94a].

While the graphs in figures 1(a - h) are primarily used to quantify SCAFS benefits, we can see that the benefits in CPU time and response time of using SCAFS is inversely propotional to the number of tuples satisfying the query. So SCAFS is most effective in this sense when the query result size is small. Compressing data does not affect the benefits due to SCAFS. The variation in the benefit of SCAFS for different file structures can be attributed to characteristics like fill factor, number of overflow pages etc.

These results provided valuable insights for the application and for the broader exploration of the potential of SCAFS. The results discussed in this section serve to emphasise the fact that the benefit of SCAFS

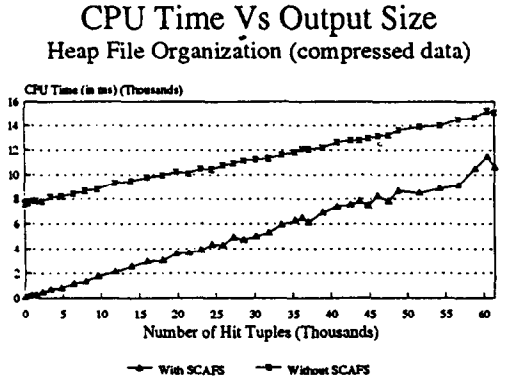

fig. 1(e)

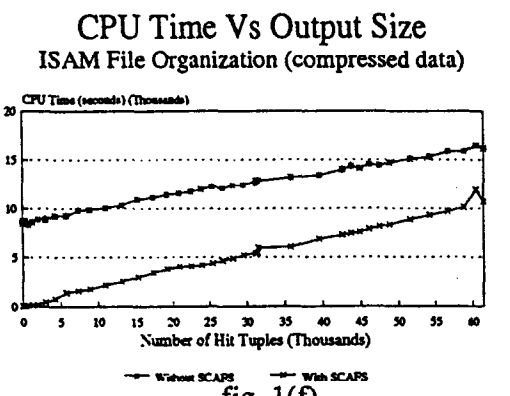

fig. 1(f)

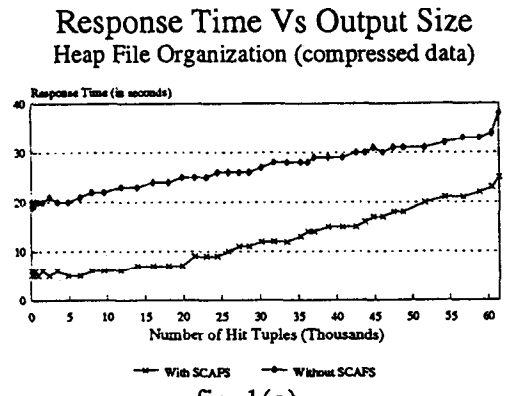

fig. 1(g)

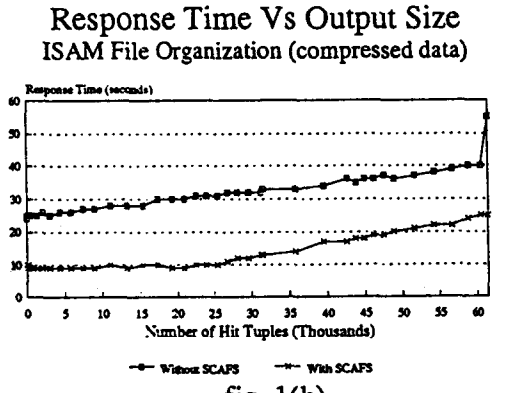

fig. 1(h)

depends on the file organization of the table being searched and great care must be taken when employing SCAFS.

4. The Effect of Scheduling on the SCAFS Board

In this section we investigate the effect of different scheduling policies on the performance of SCAFS. We first discuss how the value of the SCAFSSCHED kernel parameter effects scheduling on the SCAFS board. We then study the effect of different SCAFSSCHED values ranging from 1 to 40000.

For a small value of SCAFSSCHED, the scheduling is fairer in that all the queries get a small time on the board and are not waiting for too long a time in the queue, but the individual response times for each query increases. For very small values of SCAFSSCHED we may find that there is a large system overhead due to scheduling and the advantage of using SCAFS is lost due to this overhead.

For large values of SCAFSSCHED, queries will be waiting in the queue for long periods of time though results for the queries will start being received much sooner. For a very large value of SCAFSSCHED, we may find that it is equivalent to running the queries in serial order 3 at a time. The total time taken to receive the results of all the queries will remain the same.

These observations led us to seek more detailed insights into the effect of this parameter. For a more detailed discussion of the SCAFSSCHED parameter see [ANAN94a].

The graphs in fig. 2(a - h) show how SCAFSSCHED affects the execution on 52, 39, 26 and 13 queries on a single location table of heap file organization. As the value of SCAFSSCHED is increased the time between all the queries having started and the time the first results are received reduces. From the graph in fig. 2 (a) we see that there is not much difference in the performance of SCAFS for SCAFSSCHED values of 20000 and 40000. Thus there must exist a value for SCAFSSCHED with the

Query Response Vs Time
Query Load : 52 queries (1)

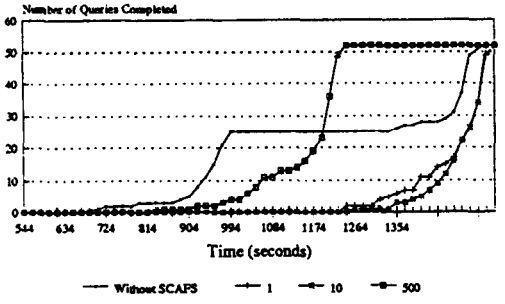

fig. 2(a)

Query Response Vs Time
Query Load : 52 queries (2)

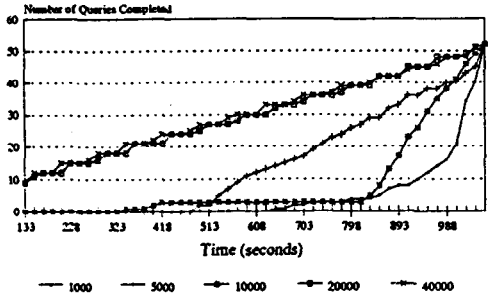

fig. 2(b)

Query Response Vs Time
Query Load : 39 queries (1)

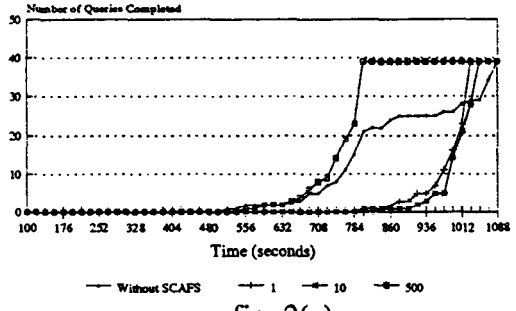

fig. 2(c)

Query Response Vs Time
Query Load : 39 queries (2)

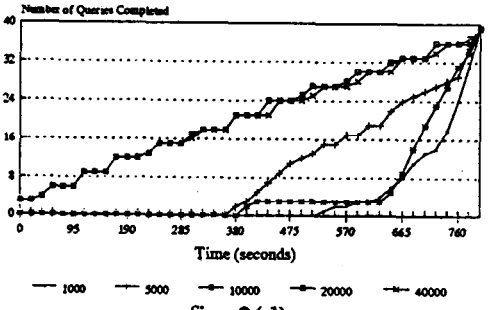

fig. 2(d)

Query Response Vs Time
Query Load : 26 queries (1)

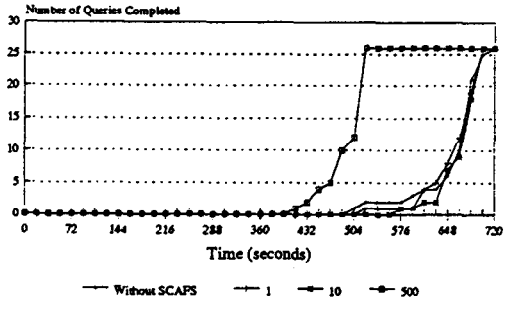

fig. 2(e)

Query Response Vs Time
Query Load : 26 queries (2)

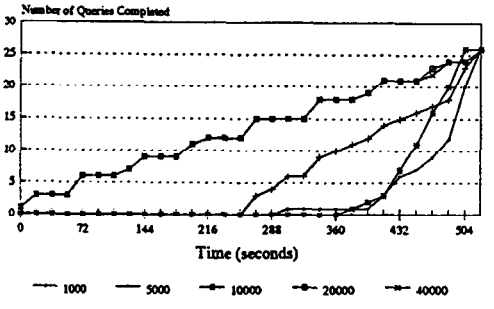

fig. 2(f)

Query Response Vs Time
Query Load : 13 queries (1)

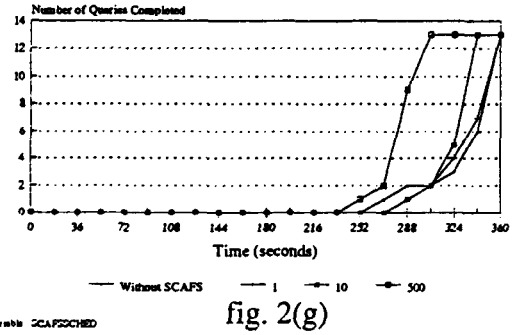

fig. 2(g)

Query Response Vs Time
Query Load : 13 queries (2)

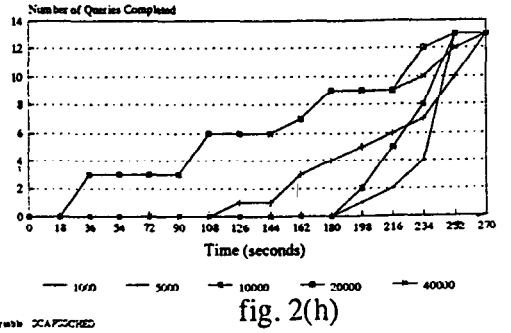

fig. 2(h)

property that there is no increase in performance for any values greater than it. This value of SCAFSSCHED is probably dependent on the time SCAFS takes to scan the table being queried i.e. the size and file organization of the table being scanned (see section 3). Also, the graphs for a constant value of SCAFSSCHED with varying query load are of very similar shapes and average query response times per query. Therefore the performance of SCAFS for a particular value of SCAFSSCHED does not seem to depend on the query load.

The graphs in fig. 3 show the total response time for a varying number of queries on a single location table of heap file organization for different values of the SCAFSSCHED parameter. As can be seen from the graph in fig. 3(b), for SCAFSSCHED values ranging from 1000 to 40000 there is no appreciable change in the total response time for the queries. However, if the value of SCAFSSCHED is reduced to less than 500 there is an increase in the total response time of the queries (fig. 3(a)). For a SCAFSSCHED value of 1 or 10 the total response times are equal to those when not using SCAFS. This degradation is due to the system being overloaded by swapping on the SCAFS board. For SCAFSSCHED value 500 the total response times are comparable to those for higher SCAFSSCHED values. Thus, increasing the value of SCAFSSCHED to a value greater than 500 has little effect on the total time taken by SCAFS to execute all the queries.

Query Load Vs Response Time
Varying SCAFSSCHED

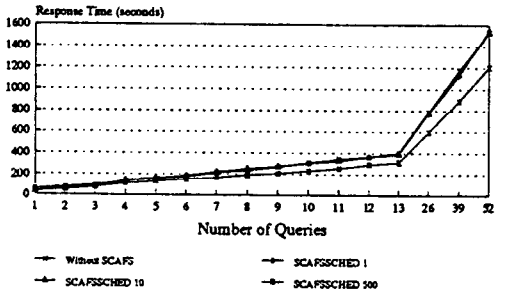

fig. 3(a)

Query Load Vs Response Time
Varying SCAFSSCHED

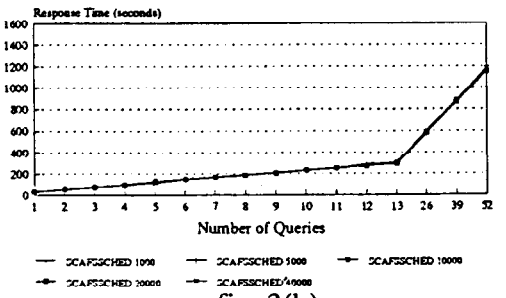

fig. 3(b)

These results show how important this parameter is in multi-user environments, such as the NIHE database system. Clearly, our experimental results, displayed in figs. 2 and 3, validate our predictions on how the SCAFSSCHED parameter would affect the performance of SCAFS and therefore, the performance of the database system.

5. The Effect of Table Locations

When more than one query is being executed at the same time on data lying on the same disk, there is disk contention which increases the execution time of the individual queries. At any point in time the maximum number of queries searching data in parallel using SCAFS is three, so the maximum disk contention occurs when all three queries are searching data on the same disk.

Disk contention can be reduced by splitting data over a number of locations. Locations could be on the same disk, the same SCSI (Small Computer System Interface) channel or separate SCSI channels. For SCAFS to be employed all the table locations must be accessible to SCAFS.

Splitting data over more than one location on the same disk does not reduce disk contention as there are still three queries searching for data on the same disk concurrently. In fact disk contention may be increased as the disk head would spend more time on seeks .

If data is split over more than one disk on the same SCSI channel, then there is a possibility that at some time the queries on the SCAFS board are searching data on separate disks, reducing the disk contention.

If data from a table is split over disks on separate SCSI channels, we increase the parallelism in the execution of the queries. The disk contention remains the same. There are now a maximum of 3*n queries that could be running simultaneously (where n is the number of SCSI channels that the data is spread over), three on each SCAFS board.

As the advantages of splitting data over different disks on the same SCSI channel, i.e. reducing disk contention, and over different SCSI channels, i.e. increasing parallelism, are independent of each other, a blend of both methods would be expected to give best results.

For this part of our performance study we looked at how the total response time varies for different work loads on tables with different numbers of locations. The results are presented in fig. 4(a - c). As can be seen in fig. 4(a), a table with two locations on separate SCSI channels gives lower response times than when the table is stored at a single location, at two locations on the same disk or at two locations on separate disks but sharing the same SCSI. The graph in fig. 4(b), shows that the response times for a table with 3 locations on separate SCSI channels are lower than that for 2 locations (fig. 4(a)) but

the response times for a table with 4 locations on separate SCSI channels are comparable to those for a table with 2 locations on separate SCSI channels. A table with four locations, two on each SCSI channel performs better but still not as well as the table with three locations on separate SCSI channels. In fig. 4(c), we see that response times for a table with 5 locations are higher than those using three locations on separate SCSI. The table with 6 locations, 2 on each SCSI, gives the lowest response times but the improvement over the 3 location table is negligible (see fig. 4(b)).

Query Load Vs Response Time

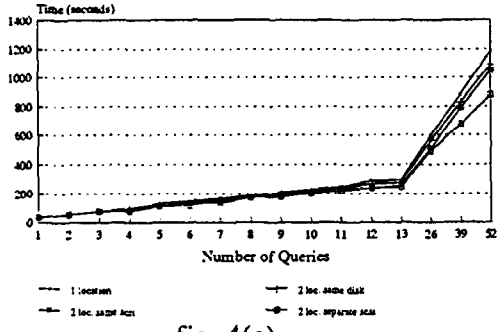

fig. 4(a)

Query Load Vs Response Time

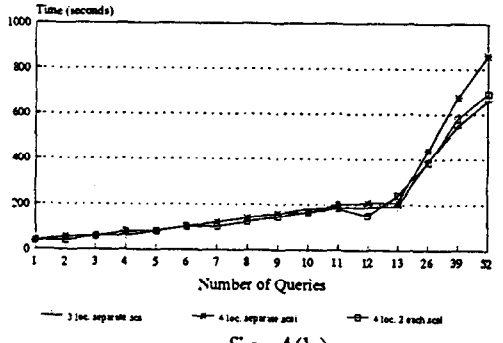

fig. 4(b)

Query Load Vs Response Time

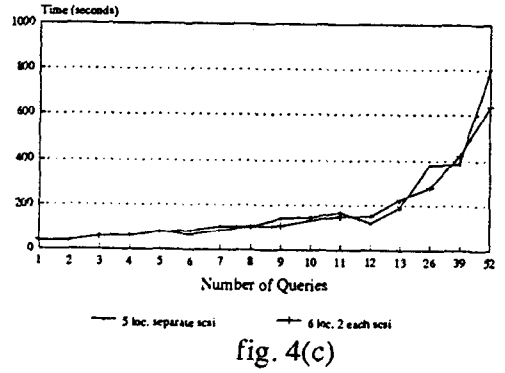

fig. 4(c)

The above results indicate that using 3 separate SCSI channels simultaneously gives the best results for our application and configuration. The reason for this could well be some constraint imposed by the hardware e.g. size of main memory. More SCSI channels being used simultaneously would mean more queries being run in parallel and an increase in memory requirements.

The above results also indicate that spreading a table over separate SCSI channels reduces response times to a much greater extent than spreading a table over separate disks on the same SCSI channel. The queries used in the test were staggered by 2 seconds each. If these queries were staggered by a longer time having two locations on the same SCSI could have more effect on response times. Also the SCAFSSCHED parameter for the tests had a fixed value of 1000 ms. For larger values of SCAFSSCHED the spreading of data over locations on the same SCSI channel may have a greater effect.

6. The Ingres Query Optimizer and The Ingres Search Accelerator

The decision of whether or not SCAFS should be used in the processing of a particular query is solely the decision of the Ingres Query Optimizer. During our investigation of this decision making process we found that at present SCAFS is not being used to its full potential by Ingres. We present here a few guidelines that have been found to be quite accurate in predicting whether SCAFS is going to be employed by the Ingres Query Optimizer.

6.1 Heap File Organization

- SCAFS is always employed for searching heap tables.

6.2 Hash File Organization

- SCAFS is used to search hash files except when the *where* clause of the query contains equality predicates on all key fields 'anded' together.
- When searching on secondary indexes on the hash table, if the expected number of 'hit' tuples is >= 13.82% of the whole table, SCAFS is employed.
- When searching on secondary indexes, if the *where* clause uses 'not like', 'not between' or 'like' with a % as the first letter in the like-pattern, SCAFS is always used.
- If the *where* clause has two predicates on secondary indexes anded together, SCAFS is employed only if
  - over 3.35% approx. of the whole table is expected as the result to the query and
  - each of the predicates, individually, expects to 'hit' 13.82% of the whole table
- If the *where* clause has an 'or', SCAFS is always

used.

6.3 ISAM File Organization

- If an ISAM file is being searched on its primary index, SCAFS is only used if the *where* clause uses a like-pattern with a % as its first character.
- When searching on secondary indexes on the ISAM table, if the expected number of 'hit' tuples is >= 9.75% of the whole table, SCAFS is employed.
- When searching on secondary indexes, if the *where* clause uses 'not like', 'not between' or 'like' with a % as the first letter in the like-pattern, SCAFS is always used.
- If the *where* clause has two predicates on secondary indexes anded together, SCAFS is employed only if
  - over 1.8% approx. of the whole table is expected as the result to the query and
  - each of the predicates, individually, expects to 'hit' 9.75% of the whole table
- If the *where* clause has an 'or', SCAFS is always used.

6.4 BTree File Organization

- If an file organized as a BTree is being searched on its primary index, SCAFS is only used if the *where* clause uses a like-pattern with a % as its first character.
- When searching on secondary indexes on the BTree, if the expected number of 'hit' tuples is >= 8.74% of the whole table, SCAFS is employed.
- When searching on secondary indexes, if the *where* clause uses 'not like', 'not between' or 'like' with a % as the first letter in the like-pattern, SCAFS is always used.
- If the *where* clause has two predicates on secondary indexes anded together, SCAFS is employed only if
  - over 1.575% approx. of the whole table is expected as the result to the query and
  - each of the predicates, individually, expects to 'hit' 8.74% of the whole table
- If the *where* clause has an 'or', SCAFS is always used.

The conclusion we arrived at after testing the Ingres Query Optimizer was that the optimizer only considers the use of SCAFS after it has made the decision to scan the table. Thus, it does not take any of the benefits of using SCAFS into account when making the decision whether SCAFS should be employed or not. This results in queries being executed without SCAFS even though it would be more efficient to use SCAFS.

7. Conclusions and Future Work

A number of conclusions and observations emerge from this study and although they should be considered anecdotal rather than scientific at this stage, we believe they give useful insights into the utilization of SCAFS for large Ingres Databases.

- The improvement in response time and CPU time brought about by SCAFS depends on the organization of the file and the number of 'hit' tuples for the query.
- The time taken by SCAFS to scan a table can be reduced by using a small index or some form of hashing that would reduce the amount of data to be scanned. Especially for large databases it would be useful to be able to use sparse indexes along with SCAFS to reduce the amount of data to be scanned without putting an undue load on the CPU. Wiles [WILE85] showed how secondary indexes could be used in conjunction with CAFS-ISP, the version of CAFS for the VME environment. Such an approach could be useful in the case of SCAFS as well.
- Tuning the SCAFSSCHED parameter for a specific application can affect the performance of the system considerably.
- The number of table locations over which the table being searched is spread affects the performance of SCAFS for high work loads and can have a considerable effect on the performance of the system.
- The Ingres Query Optimizer does not take into account the full power of SCAFS when deciding when SCAFS should be used to search an Ingres table.

The results can also be used to create a kind of query pre-processor that can rewrite a query into a semantically equivalent query that forces the Ingres Query Optimizer to process the query using SCAFS. Such a pre-processor linked with Ingres could have great potential for use in high frequency on-line transaction processing database applications that are CPU bound [ANAN93,ANAN94].

A fairly simple tool that predicts the optimal value of the SCAFS scheduling parameter and the number of table locations over which the table being searched should be spread given a work load could be useful at database design time.

We are working on these issues at present and also recommend further research into the use of secondary access paths for associative memories. Earlier studies into the use of secondary indexes in conjunction with CAFS-ISP [WILE85] have shown that there is benefit in using

secondary indexes on ISAM files in conjunction with CAFS.

We believe that these results will provide Ingres Database designers with useful insights into the prediction and improvement of the performance of databases being developed.

8. Acknowledgements

We would like to thank the ICL SCAFS team at Bracknell and the members of the project team at the University of Ulster and the Northern Ireland Housing Executive for their help and valuable suggestions in performing the above tests on SCAFS. The research was funded by the Industrial Research and Technology Unit, Northern Ireland and ICL Ltd. and Kainos Software Ltd. were the industrial partners.

9. References

[ANAN93] S.S. Anand, D.A. Bell, J.G. Hughes. Using Semantic Knowledge to Enhance the Performance of Specialized Database Hardware. Workshop of the FSTTCS'93 Conference, IIT Bombay, December, 1993.

[ANAN94] S.S. Anand, D.A. Bell, J.G. Hughes. Database Mining in the Architecture of a Semantic Pre-processor for State-Aware Query Optimization. Proc. of the AAAI-94 Workshop on Knowledge Discovery in Databases, Seattle, July, 1994.

[ANAN94a] S.S. Anand, D.A. Bell, J.G. Hughes. Performance Enhancement using SCAFS in a Large, Transaction Intensive Database System. To appear in Informatics Research Reports Issue No. 9, University of Ulster, Oct. 94.

[BABB79] Ed Babb. Implementing a Relational Database by means of Specialized Hardware. ACM Transactions of Database Systems, Vol. 4, No. 1, March 1979.

[BABB85] Ed Babb. CAFS file - correlation unit. ICL Technical Journal, Nov.. 1985.

[COUL72] G.F. Coularis, J.M. Evans, R.W. Mitchell. Towards content addressing in databases. The Computer Journal, Vol. 15, No. 2, 1972.

[INGR91] The Ingres Search Accelerator User's Guide

[LEUN85] C.H.C Leung, K.S. Wong. File processing efficiency on the context addressable file store. Proc. VLDB Conference, 1985.

[NIHE88] Property, Rent Accounting, Waiting List - Operational Requirements Vol. 1 - Statement of User Requirements.

[WILE85] P.R. Wiles. Using Secondary Indexes for Large CAFS Databases. ICL Technical Journal, Nov. 1985.