P O L I S H I N G T H E T A R N I S H E D I M A G E

A brief study of the effects of some user available options

Brian Duncombe

Carolian Systems, Inc.
Richmond, Ontario

POLISHING THE TARNISHED IMAGE

A brief study of the effects of some user available options

NOWRUG'84

Seattle, WA

May 16, 1984

Brian Duncombe

Carolian Systems Inc.

Richmond Hill, Ontario

(416) 498-1515

In my capacity as a consultant in the effective use of the HP3000 computer system, I often observe that users are not making the most effective use of the tools available to them.

A case in point is the use of various processing options that are available to the IMAGE/3000 user. We often hear users commenting on the lack of features available within IMAGE/3000 (notably the lack of multi-threading access) yet I seldom see all of the features that are there being used to the greatest advantage.

What I am presenting here is the results of some simple experiments that I conducted to show the relative benefits of the various techniques available with the IMAGE/3000 environment.

The things that I tried are as follows:

- varying buffer specifications for the data base
- using the output deferred processing mode
- pre-sorting transactions prior to updating the database
- enabling disc caching

WARNING

The results presented here are for a very simple database with very simple updating (additions only) and the processing was done on a stand-alone computer system. While I believe that the results are not representative, I do believe that the relative effects are valid. My experiences in the past have yielded very similar results with actual application systems but not necessarily quite so dramatic improvements.

HARDWARE ENVIRONMENT

. HP3000 Series 44
. 1024 KB main memory
. 2 HP7925M disc drives on 1 GIC
. stand-alone system in all cases

SOFTWARE ENVIRONMENT

. MPE 'P' (pre-release version)
. SPL (test programs)
. IMAGE/3000
. NO transaction logging

. NO intrinsic level recovery (ILR)

Figure 1

DATABASE USED

```
BEGIN DATA BASE DBTEST;
PASSWORDS:
ITEMS:
     KEY1,              X 4;
     KEY2,              X 4;
     KEY3,              X 4;
     X02A,              X 2;
     X02B,              X 2;
     FILL01,            X100;
     FILL02,            X100;
     FILL03,            X100;
     FILL04,            X100;
SETS:
NAME:   M-PROD,         AUTOMATIC ;
ENTRY:  KEY1           (1);
CAPACITY: 2001;
NAME:   A-2,            AUTOMATIC ;
ENTPY:  KEY2           (1);
CAPACITY: 201;
NAME:   A-3,            AUTOMATIC ;
ENTRY:  KEY3           (1);
CAPACITY: 101;
NAME:   D-INVENT,       DETAIL    ;
ENTRY:  KEY1           (!M-PROD          ),
        X02A,
        KEY2            ( A-2),
        X02B,
        KEY3            ( A-3),
        FILL01,
        FILL02,
        FILL03,
        FILL04;
CAPACITY: 12000;
END.
```

Figure 2

OVERVIEW AND INTRODUCTORY COMMENTS

As you can see from Figure 2, the database is very elementary. Since the master data sets are all automatics, the media record size is very short and makes it easier than normal for the IMAGE software to manage additions and references to these data. I did attempt to fill out the detail somewhat but even this is not representative of most data base designs. The structure does; however, make it fairly simple to test the effects of various techniques and thus see generalized trends.

To control the experiments, I built a system on a set of packs to be used exclusively for this testing. Once I had the data files created, I simply erased them between tests to eliminate any effects of file placement. I did not make any effort to position the files optimally but rather let MPE distribute extents with default assignments.

Throughout the testing, all reported times are elapsed times rather than processor (CPU) times. Since this is the case, it is possible that parts of the results have been skewed by the processor being fully utilized and that with a faster processor, the relationships between the results might be slightly distorted. As an aside, I did try to watch for this and there were no major portions of the processing where the processor was saturated except during the caching tests when the caching software tends to utilize most idle processor cycles.

The transaction file was generated using a random number generator to produce the three key values per transaction. In doing this, I get what is probably the worst case transaction sequence since in normal application systems, there is usually some 'clumping' of the key values for an order, invoice, customer, etc.

Although the facilities were available to me, I chose not to enable either transaction logging or intrinsic level recovery (ILR) for the tests. This was initially done to reduce the number of permutations and combinations for the testing and reporting. I did; however, repeat several of the tests to see what the effect of enabling these facilities would be. Running with these facilities enabled showed a slight increase in elapsed times but the increase was very slight and not worth reporting in detail.

THE EFFECT OF 'BUFFSPECS'

The most obvious technique that is available to IMAGE/3000 users is that of over-riding the default buffer specification for the database. It seems that the people who maintain IMAGE/3000 have either never thought about choosing a good default specification or else they thought about it years ago when a 320KB system was a monster. The nice thing about this parameter is that it is easy to change and requires no program changes. The only thing you must do is to sign on to the system as the creator of the database (or have and know the maintenance word) and get exclusive control of the database using DBUTIL.PUB.SYS to respecify the BUFFSPECS parameter.

The following chart (Figure 3) shows the effect of varying the buffer specification throughout a wide range of values. Subsequent charts will always show the effect of varying this BUFFSPECS parameter in addition to whatever else is under consideration.

Conclusions:

* By simply over-riding the default buffer specifications, you can gain about 22% in the run time of the job.
* The gain is most noticable in the lower range of values.
* Assuming that you are not already short of main memory, you will almost always enjoy some gains.

Warning

* For batch processing, this will almost always be a good idea but for environments where many databases are being used by a few people each you should not go overboard with the BUFFSPECS. If you increase the specifications to the point where the size of the Global Control Block starts to cause main memory resource shortages, the effect of having a big buffer pool is outweighed by the occurence of the control block being swapped out of memory. If this is the type of database use during the daytime only, you might consider modifying the specifications at the beginning of the evening batch processing and resetting them for the next daytime period.

INCREASING BUFFSPECS

Random Sequence

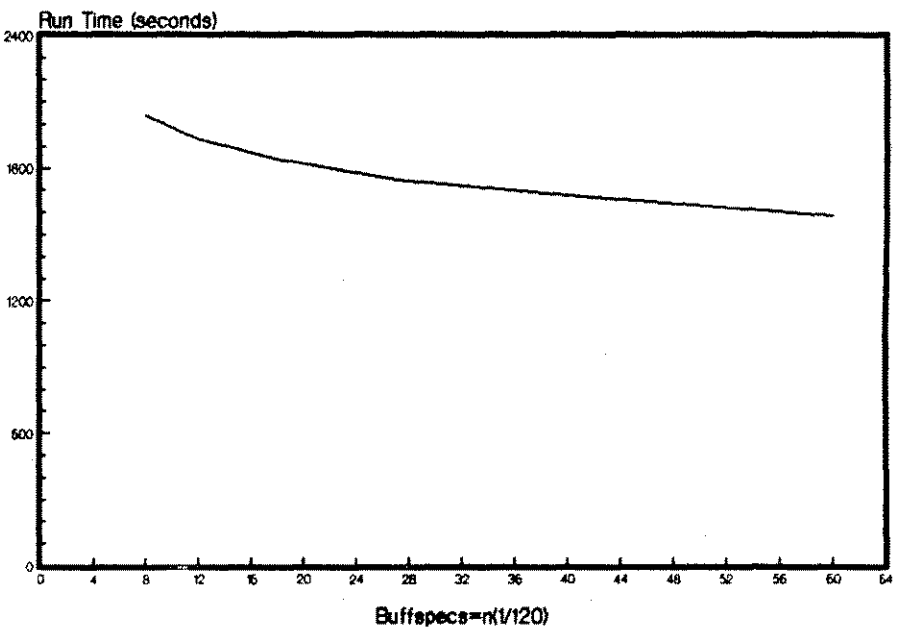

Figure 3

THE EFFECT OF USING OUTPUT=DEFERRED

A feature of IMAGE/3000 that is overlooked by many users is that of telling IMAGE to defer physical disc writes until absolutely necessary. This feature in effect turns off the feature in IMAGE that ensures that the copy of the database on disc is updated almost immediately upon a user program issuing an intrinsic that modifies the data. Because of the nature of this facility, there are several conditions which must be met. In the first place, you must be the only one currently accessing the database and to ensure this, IMAGE forces you to open the database in MODE=3. In the second place, you would be well advised to be backed up prior to commencing processing in this mode since in the event of an operating system failure, you are almost guaranteed of having a corrupt database. The easiest way of doing this is to schedule your daily backup to a period between the end of daily multi-user access and the beginning of nightly batch processing. Once this is done, you know you are well backed up and you can then enjoy the benefits of this technique for your batch processing. Because of the nature of the processing, you should turn off transaction logging and intrinsic level recovery (ILR). Once these conditions have been met, it is a simple matter of calling DBCONTROL with a mode of '1' to set the deferred output mode. Of course, in order to use this technique, you must make some changes to your application code. These changes may be very simple or they might be so complex as to not warrant the effort.

The following chart (Figure 4) shows the effects of utilizing this facility compared to the results without it. As you can see, the gains are quite attractive. In addition, notice that the improvements are more significant as the number of buffers provided is increased. This seems logical since the more space you provide, the more likely a record is to be still in a buffer when it is required for another request. My experience is that most users will cut their run times in half when they can combine increased buffers with deferred output.

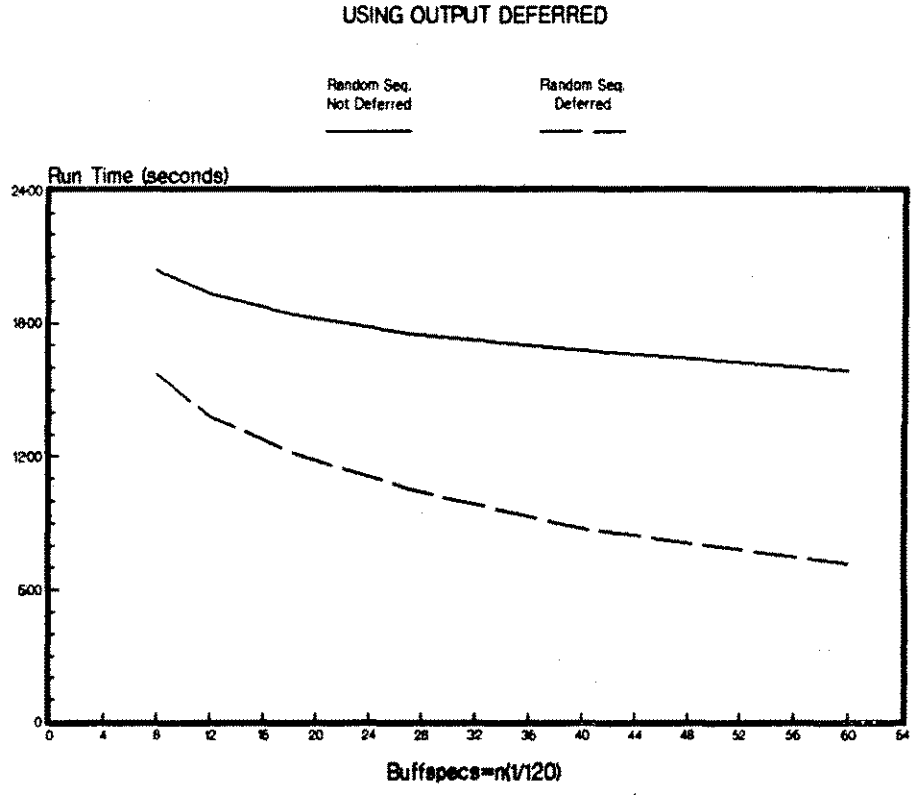

Figure 4

THE EFFECT OF SORTING THE TRANSACTIONS

The techniques that many of us learned in the earlier days of batch-only processing seem to have been discarded with the dawn of 'modern' data processing. One thing that most of us were forced to do was to sort transaction files into sequence by a key and then process those records against the 'master' file. This was necessitated by a lack of any (or at least an adequate) random access facility to our data. If we were to go back to the good old days and re-learn some of these lessons, many of us would be much better off.

The theory behind this technique is that if you have several transactions that must be processed against the same master record, it will be better to get the master record once and then do all the processing rather than pay the price of accessing the master individually for each transaction as it comes along.

In order to impliment this sorting, I modified the original program to include an 'internal' sort which was included in the run times. If the sort were done as a seperate step, the gains would be somewhat reduced.

Of course, this technique will likely modify the logic of your current applications and as such may be difficult or totally impractical to impliment. Even if you cannot use it retroactively, the following results should stir you to keep it in mind for future design/development projects.

The next chart (Figure 5) shows the possible benefits of this seemingly simple idea. As you can see, the benefits are worth having if the cost of implimentation is not too great. Notice that the benefits are most noticable when combined with low buffer specifications and that fairly early in the chart, the effect of buffers is almost negligible. This is of course because in presorting the transactions, you have greatly increased the chances of a needed record being present from some previous transaction (the last one!). Because of this, we can conclude that the default buffer specification is less harmful when you have sorted the data (HP gives you credit again?). If the database design and application were more complex, this point of no additional improvement would move out to the right as it should if multiple users were sharing the database simultaneously.

Note:

> In an investigation for a client, I replaced the sort on the key value with a pre-calculation of the hashed key value and sorted on this hash key. The theory was that by doing this, I could reduce the movement of the disc arm mechanism by requesting addresses that were as close as possible. In fact, with the transactions used in this test, I could not detect any significant difference in the run times although I have seen other applications of pre-calculation of hash keys that yielded very favourable improvments.

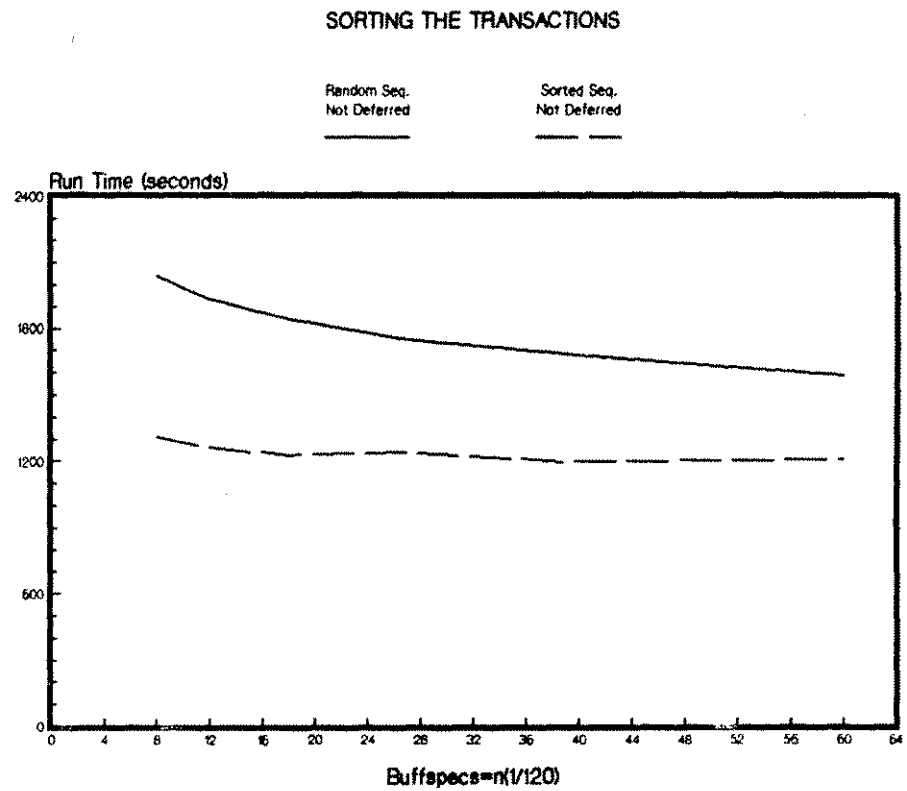

Figure 5

THE EFFECT OF COMBINING BUFFERS, DEFERRED AND SORTING

The next logical experiment is to combine the variables examined so far. Figure 6 shows the results of this combined effort.

As you can clearly see, if you are able to apply all three ideas, the improvment in run times is about five fold. My experience to date is that this is optimistic but even a three fold improvment would extend the life of your system greatly, allow more applications to be run on the same system or reduce the need for additional shifts of operators. In some cases, it can mean the ability to be finished yesterdays work by the time you arrive today.

Most installations are not able to combine all three changes and so to some extent this chart shows an un-attainable goal. It also shows that IMAGE/3000 is not deserving of much of the criticism it has been receiving lately. If we were to employ some fairly simple principles in our system design and implimentation, we would be repaid hansomely. Of course, it would be nice if we could buy our way out of our difficulties but currently that is not an alternative for many of the larger installations.

COMBINING SORTING WITH OUTPUT DEFERRED

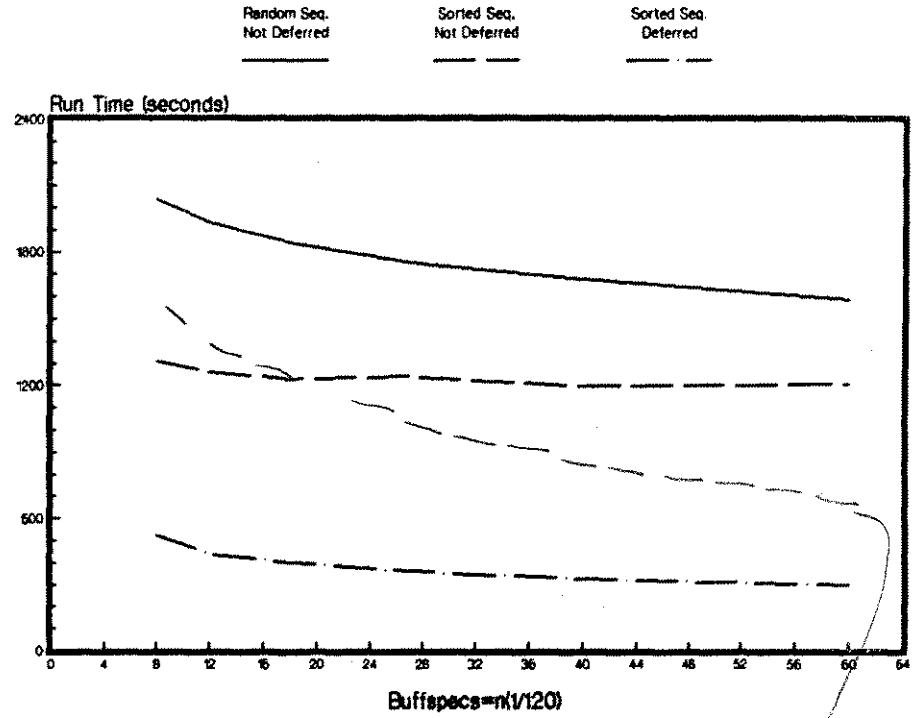

Figure 6

RANDOM
Deferred

THE EFFECT OF BUFFSPECS COMBINED WITH CACHING

The results presented here have been determined by running the same set of tests as reported earlier but on a system with disc caching enabled. In fact, the tests were all conducted on the same system with the same version of operating system (MPE5P). The previously reported results were those obtained with caching turned off (not turned on).

The following results are those obtained after caching was turned on. One comment on these tests is that MPE5P is probably the optimal version since it enjoys the benefits of disc caching without the performance degradation of the new operating system (MPE5E) and it's logic for extended table sizes. A look back at Figure 1 will show you that the main memory size (1 MB) was not up to the recommended specifications. The application that I was running made very low demands on main memory resources itself although the results reported here should be able to be improved upon somewhat with additional memory.

Due to a lack of hardware resources (disappearance), the testing with caching was only carried out with buffer specifications up to 27 buffers. Some of the charts will lead you to wonder how the extended caching results might have compared but we will probably never know.

Figure 7 shows the baseline test plus the same test with caching. These initial results are much better than either those advertised or my own experiences. As I said earlier, these results should be used to compare relative effects and should in general be considered optimistic. As you can see, increasing buffer specifications has much less effect when caching is turned on since caching effectively extends the available buffer space through the caching mechanism. What is not reported here and something that is significant is the CPU resource consumption data for the tests. In general, the Series 44 processor was fully utilized for most of the caching tests. If the transactions had been more CPU intensive themselves, the caching results might not have been nearly as appealing. My personal experiences with caching are that if you can get the same effect by some other technique (buffering, blocking, sorting, etc.) you will always be better off than incurring the caching overhead although where this cannot be done, caching is an extremely worthwhile facility.

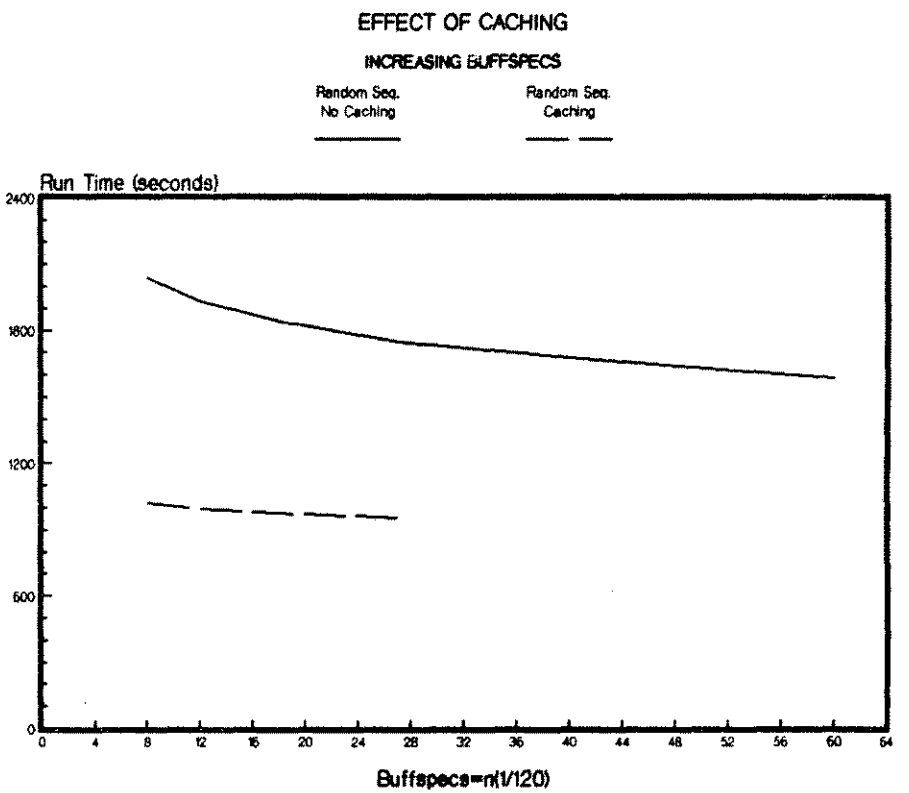

Figure 7

THE EFFECT OF COMBINING DEFERRED OUTPUT WITH CACHING

Figure 8 repeats the results shown earlier (Figure 4) and in addition repeats those two tests with disc caching enabled. As you might expect, caching again shows its best results with low buffer specifications. The addition of output deferred to the caching still yields significant savings although not nearly as much as when caching is not present. This appears to be logical since both deferring output and caching are attempting to reduce the loading on the disc subsystem and their effects overlap one another.

If we can attempt to extrapolate the caching data, it would appear as if the test results for deferred output with large buffer specifications might very well be better than caching without output deferred. In effect, a user who takes advantage of current tools may very likely get better performance than one who relies on disc caching as his/her salvation. The results when all three techniques are employed still appears to be an improvment over any other combination seen so far.

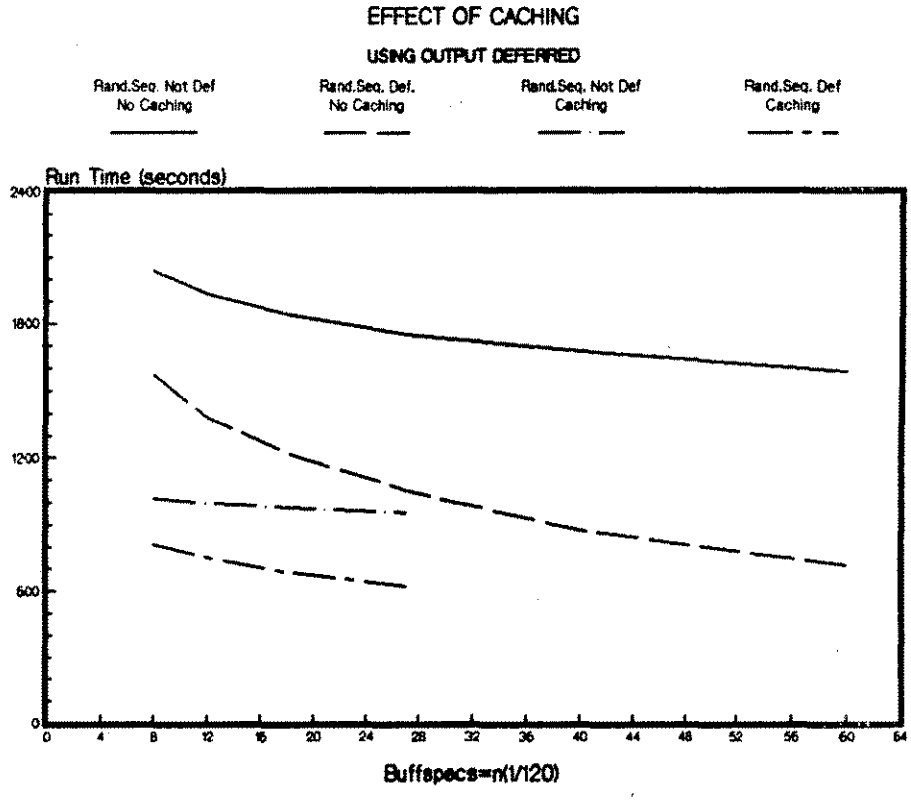

Figure 8

THE EFFECT OF COMBINING SORTING WITH CACHING

As Figure 9 shows, the improvments see when disc caching is introduced appear to be as significant as those obtained by sorting alone. In combination, the results are very attractive.

Thinking about this case, it would seem logical that many of the advantages attributed to the increased size of main memory used to store disc data that disc caching offers are of no use when the transactions are being sorted prior to processing against the database. Caching will still yield run time reductions due to the 'instantaneous' completion of disc writes rather than the waits that are imposed by IMAGE usually.

If the database were being shared by multiple users, one would expect that sorting would be less effective by itself since other users would cause the buffers to be reused even when the test transactions referenced the same data repeatedly. In this case, the extended main memory that caching provides for buffering space might very well be more significant. In the same vein, what appears to be no improvement when buffspecs are increased will probably become a noticable improvement when the database is accessed by multiple concurrent users and will be more effective than leaving the task to caching.

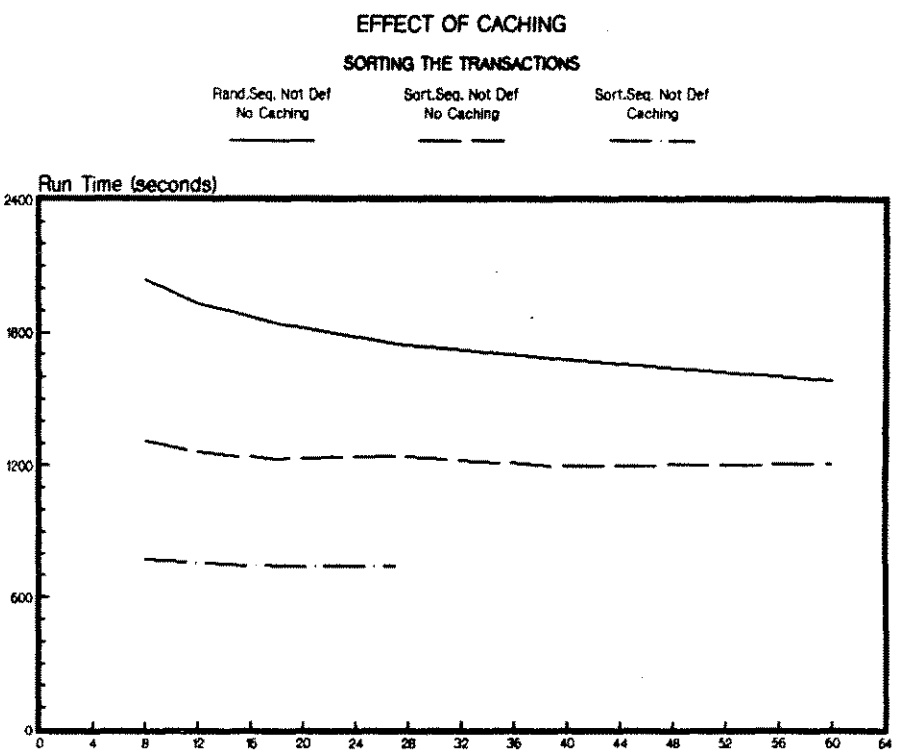

Figure 9

THE EFFECT OF COMBINING ALL TECHNIQUES

As we have been going along, we have been seeing a steady improvment in run times as each technique was employed. Figure 10 compares some of the previous tests with the final one that uses sorted transactions, increased buffer specifications, deferred output and disc caching. The chart clearly shows that the utilization of all four techniques together will yield the best elapsed time for this particular run environment. What is also notable is that the second best result is obtained by turning caching off. When the other three tools are utilized, the addition of caching is not nearly as significant as the addition of most any other technique in the other tests. This seems reasonable since caching is attempting to do with additional overhead many of the same things that the already available tools have been capable of providing all along.

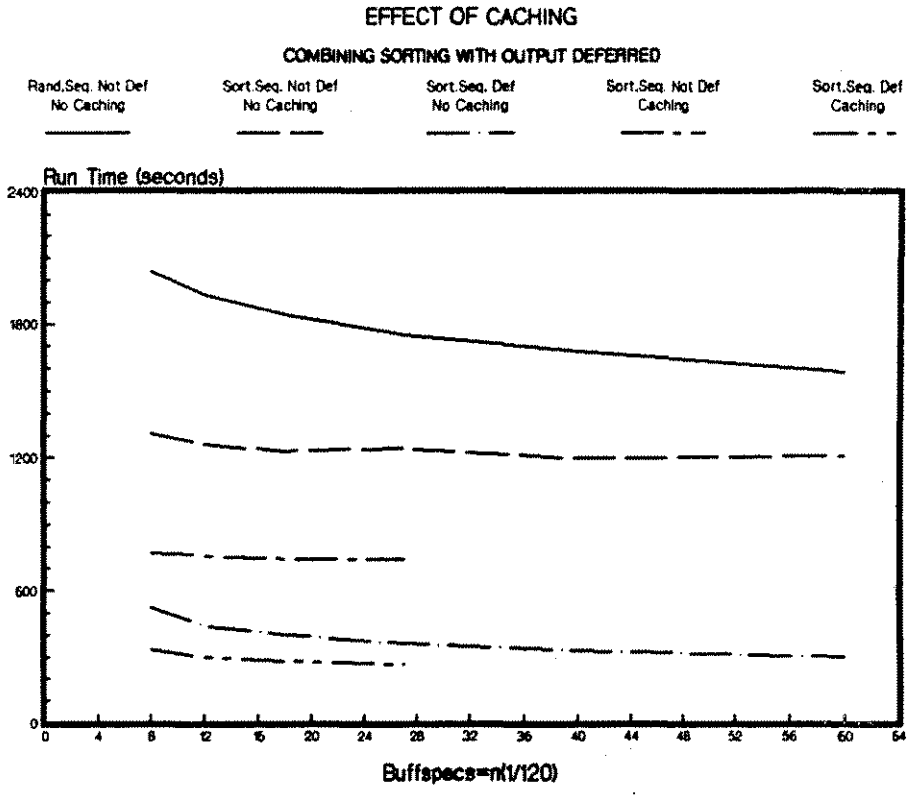

Figure 10

SUMMARY

Unless main memory is at a premium on a system, it is always advantageous to over-ride the default buffer specifications for a database. This should of course be tempered with common sense. It is not advisable to run off and change the specifications of 20 or 30 databases if there is a good chance that they will all be opened at the same time for normal processing. The addition of this increased main memory requirement can be significant even to a system with a very large main memory configuration.

Where possible, modifying programs and job schedules where necessary to include processing the data base with deferred output to disc will always produce significant improvements in run times. In addition, by reducing the number of times physical disc i/o must be requested you reduce the CPU usage marginally since the code to handle this function is executed less often. This is; of course, always weighed against the effort to impliment it and the effect that the restrictions imposed by this facility will have on your processing in general.

The pre-sorting of transactions has a clear possibility of improving overall run times. Of course, there are many times when the effort expended in going back to change existing applications exceeds the benefit. This is; however, a very useful technique to keep in mind for future applications where it can be designed in. The choice of what to sort on requires thought on your part and quite possibly some testing.

While disc caching always appears to improve the situation in the results reported here, my experience has shown me that there are many cases where the CPU does not have the excess processing capability to absorb the overhead of caching and that many applications will run slower with caching. I would recommend that you aquire the services of a consultant with considerable experience in disc caching (HP or third party) if you are in doubt as to whether your system would benefit from disc caching.