HEWLETT-PACKARD GENERAL SYSTEMS USERS GROUP

FEBRUARY, 1980 MEETING

RESOURCE OPTIMIZATION SERIES

OPTIMIZING IMAGE: AN INTRODUCTION

Rick Bergquist

American Management Systems
561 Pilgrim Drive, Suite D
Foster City, California 94404
(415) 573-9481

ABSTRACT

For installations using IMAGE as the mainstay of their data requirement needs, optimizing the factors affecting IMAGE's performance can have a drastic effect upon response time and system throughput. The factors affecting IMAGE performance are reviewed within this paper along with suggestions on how to minimize resource usage. Also, the report produced by the program DBLOADNG is examined to show potential IMAGE problems which can be isolated.

INTRODUCTION

IMAGE, like all programs, performs tasks by consuming computer resources such as CPU, channel, main memory, and disc. The rate of resource utilization varies depending on the particular task and on the configuration of the system. The purpose of examining IMAGE is to determine how to increase the efficiency with which it uses system resources.

This may be accomplished in two ways. Overall resource usage can be minimized by assuring that tasks are not performed more often than necessary. For example, consider the steps involved in paying bills. One does not go to the bank and open a new checking account to write each check. Instead, a single checking account is opened and many checks are processed against it. Similarly, in using IMAGE, we wish to take advantage of the fact that some tasks need not be repeated.

A second way to minimize resource usage is to use more efficient alternate resources. For example, to minimize trips to the grocery store, we store frequently used commodities at home. This trades off the cost of multiple trips to the store against storage costs. Similar tradeoffs exist in an IMAGE system.

This paper examines resources used by IMAGE, points out methods to determine what problems exist, and offers techniques to minimize IMAGE resource utilization. Also presented are areas where no clear solutions to tradeoffs exist, and where future work might profitably be directed.

The resources of an HP3000 system which are of interest are: CPU, channel, memory and disc. Each HP3000 system varies its utilization of each of these resources, but one eventually serves as a bottleneck preventing other resources from being used. The bottleneck points for some of these resources are shown below:

The channel limits disc access to approximately 30 accesses per second. Under optimal conditions of sequential file accesses, a maximum of 58 transfers per second may be achieved. For an active system doing many unrelated disc access, only 20 to 29 accesses may occur. The 30 access per second maximum includes the disc activity required for swapping. [1]

The memory manager was designed to optimally manage systems with a small amount of memory. When memory size approaches 1.5 megabytes, or more, an excessive number of 'anticipatory writes' tends to clog the I/O channel with unnecessary write requests. System throughput can actually go down because of the unnecessary disc I/O.

IMAGE RESOURCE USAGE

CPU

IMAGE uses the CPU to verify data set field lists whenever a record is retrieved, added or updated. The validation of field lists to verify access rights can consume a large amount of CPU time. Whenever possible, an IMAGE field list should be specified once and all future references should specify the asterisk ("*") data item list. The asterisk list eliminates the need for IMAGE to verify the list each time an IMAGE call is made. Considerable savings can be obtained if a data set is being frequently accessed with the same field list.

DISC

IMAGE data sets are allocated disc space when a data base is created by running DBUTIL. Enough disc space is allocated to fulfill the capacity requirements specified in the data base schema and all disc extents are allocated when the data set is created. For master data sets, the file must be allocated in full since records are added at random locations depending on the hash. For detail data sets, record entries are appended to the data set whenever no free entries exist. IMAGE obtains all of the extents at data set creation time so it will not be unable to obtain disc space if another extent is needed.

Most data sets are expected to grow as a business grows larger, business conditions change, or data collection becomes more effective. Often, the direction of change cannot be foreseen. To be ready for change, IMAGE data sets should be sized to accomodate growth. Problems can arise when anticipated sizes for several data sets exceed the data space available. It is very likely that not all of the excess capacity in all of the data sets will be used. How nice it would be if IMAGE would be enhanced to allow for spare extents in detail data sets. Until IMAGE is enhanced, constant review is required to see that capacities are ample.

When faced with the problem of growing data entries, one answer is simply to allocate data sets with generous capacities. Unfortunately this has an undesirable affect on response time. The major factor in accessing data from disc is the amount of time required for the disc access arms to be positioned over the correct disc cylinder. As more disc space is allocated, more cylinders are taken up and access times become slower. To minimize access time, data set capacities should not be specified too generously. This is another case where detail data sets with spare extents would be helpful.

When adding or deleting entries from a detail data set, IMAGE must access the chain head entries in the associated master data sets. To minimize head movement and maximize disc throughput, master and detail data sets should, if possible, be located on separate disc drives. Similarly, if a system contains a limited number of very active data sets, attempts should be made to place the active data sets on separate disc units.

When placing data sets on separate disc drives, the loads for each disc should be examined. In typical systems the system disc (SYSDISC or device 1) receives more than its share of usage. SYSDISC contains the system file directory and swapping area as well as any user files placed on it. Swapping and file directory lookups are high access items and no extra loads should be added to them. So that no extra load is placed on the system disc, drive 0 should not be placed in the device groups DISC or SPOOL.

The placement of data sets can be accomplished, somewhat tryingly, by using RESTORE. RESTORE can be used to restore specific data sets on a specific device. The proper data set name to restore can be determined by using the table created by the schema processor. Data sets are numbered from 1 through 99 in the order listed in the schema table.

The program product ADAGER allows data sets to be placed on specific devices without the trouble of looking up data set numbers and using RESTORE. [2]

To determine which data sets to place on separate disc drives, the system should be reviewed to determine the most active programs and in turn, the most active data sets. Simple experimentation and benchmarks should show the effects of data set separation.

DISC and MEMORY

Another factor affecting disc performance is the blocking factor of data sets. By selecting a large block size, each disc access will retrieve a larger number of records. For sequential access or with master data sets with many synonyms, this can minimize disc accesses significantly. However, for random access the extra records fetched are often of little value and a penalty of extra memory space usage is incurred.

As stated in the IMAGE manual, programs which run as batch jobs without other competing processes can benefit from a large blocking factor. Programs which are run by many users concurrently are better off with smaller blocking factors. With smaller buffers the memory required for data set buffers is smaller and main memory demand is lessened. The smaller demand for memory often decreases the swapping load and increases system throughput.

A topic very much related to block size selection is the selection of the number of buffers contained in the Global Data Base Control Block (DBCB). Optimally, a sufficient number of buffers will be available so each user can have their active records kept in the DBCB thus minimizing disc accesses.

The size of the DBCB, a priveleged system extra data segment, is constrained by the size of data segments allowed on a system. The maximum size the DBCB may ever obtain is 32K words. The number of buffers contained in the DBCB is a function of the number of users accessing a data base, and the buffer specifications set by DBUTIL.

As the number of users goes up, or the number of data sets accessed becomes large, buffer demand may exceed supply and extra disc accesses will be required. To optimize the number of buffers available, IMAGE allows the number of buffers allocated to vary depending upon the number of users accessing a data base. Depending upon the number of data sets accessed by each program, the default value of allocated buffers can be modified using DBUTIL.

By specifying a larger number of buffers, disc accesses may be saved by having more buffers available. However, the advantage of extra buffers must be weighed against the additional memory requirements of the enlarged DBCB. The enlarged DBCB may force more segment swapping and an increased load upon the memory manager. Until such time as buffer hit ratios are available, analysis of the system combined with experimentation and benchmarking are the only methods available to determine the best number of buffers to allocate.

It should be remembered when selecting a blocking factor that the maximum number of buffers allowed in a DBCB is determined by the blocking factor.

Each buffer must be large enough to contain the largest block, thus the maximum buffers allowed, regardless of the specifications set by DBUTIL, is the maximum data set size minus the fixed DBCB area overhead required for locking, etc. divided by the maximum blocking factor. The maximum buffers that can fit in a DBCB may be smaller than those specified by using DBUTIL, but no warning will be issued.

Another problem associated with buffers in the DBCB is the possibility of the buffers being flushed because of a poor buffer allocation scheme utilized by IMAGE. The buffers in the DBCB are allocated in a Least Recently Used manner. When a DBPUT occurs to a detail, the detail record, its forward and backward brothers for each path, and the master records associated with the search item must be routed through the DBCB in order to update all of the required IMAGE pointers. This action may flush all other user buffers. Similarly, a long lookup under program control can also flush all of the DBCB buffers. If applications exist which will tend to flush the buffer

pool, increasing the number of buffers available may result in a net decrease in throughput since the extra buffers only expand the size of the DBCB and do not contribute to any savings in disc accesses since their contents are always flushed.

PRIMARY PATHS

The schema processor and the IMAGE procedures accept one path into a detail data set as the 'primary path'. The question arises: Of what value is a primary path? The most obvious benefit is that the primary path is the default path into a detail data set if no call has been made to DBFIND. The intent of the primary path is to allow a data set to be reorganized such that all of the records for each search item on a primary path are physically grouped together. Subsequent accesses to the reorganized data set by the primary path should be more efficient since the related records will probably be located in the same disc block, and even if in different blocks, the blocks will be adjacent.

To take advantage of primary paths, a data base must be unloaded periodicaly using DBUNLOAD,CHAINED and then reloaded with DBLOAD. Primary paths with long chains will show the greatest improvement. The improvement in data accessing time must be evaluated for each application along with the time required for the reorganization.

Often the time required for a data base unload and load will eliminate this technique from consideration. Will ADAGER be enhanced to optimize primary path record placement?

DBLOADNG and IMAGE

Given that we know the problems IMAGE is subject to, we can isolate the problems and take corrective action. The report produced by the program DBLOADNG is an aid for isolating IMAGE problems.

DBLOADNG is a program designed to review a data base to isolate potential problems. Information on a data base is obtained by traversing each data set. The information gathered by DBLOADNG is a snapshot of a data base at a particular instant. Dynamic informatiion such as buffer hit ratios or activity against each data set is not recorded. The advantage of recording information at one point in time is that the overhead of continual statistics recording is avoided. On the other hand, not all of the information available with continual monitoring is available. DBLOADNG provides no information on how the data base reached its current state. Only analysis of the transactions against a data base will tell how the data base reached its current state.

A review of the report produced by DBLOADNG is presented with a discussion of the necessary analysis and possible conclusions.

MASTER DATA SETS

By examining data set capacities and the number or entries in each data set, poor disc usage or overcrowded data sets can be spotted. Data sets which use only a small percentage of the allocated space are candidates for lower capacities.

Similarly, data sets which have a high loading factor might be candidates for additional capacity. If a loading factor is appraching 100% then spare entries are needed in order to prevent the crisis situation of no available capacity.

Within master data sets, IMAGE links synonym entries via a doubly linked list. Each primary entry is linked to a secondary entry and secondaries are linked to other secondaries via forward and backward pointers. The percentage of secondaries is shown to affect response time; the fewer the synonyms, the faster the response.

When accessing or creating entries in master data sets, IMAGE uses a hashing mechanism to locate the primary entry location. If necessary, the synonym chain is traversed to obtain the correct record. If the secondary entries for a chain are located in the same disc block then no additional disc accesses are necessary. If the synonym record is in another block then a disc access is required to fetch the synonym. Optimally, all of the synonym records in a chain will occur in the same disc block as the primary entry. IMAGE attempts to see that this occurs by placing secondary entries in the first available record slot after the primary entry. However, the fuller a data set becomes or the fewer records per block, the less likely that secondary records will occur in the same block.

DBLOADNG examines all chains and shows the percentage of pointers which point to different blocks. These pointers are referred to as inefficient pointers. If the percentage of inefficient pointers is very high, the situation can be helped by expanding the data set so entries will spread out and the number of secondaries will be reduced. The higher the percentage of inefficient entries, the slower the access time.

Data sets may have similar loading factors and similar secondary percentages, but the inefficient pointer percentages may be radically different. The different inefficient pointer percentages may stem from different blocking factors or clustering of data values. The blocking factor for data sets is listed for comparison purposes.

To show when data clustering may be taking place, the highest number of contiguous full blocks is shown. If this number is small (1-10), the record entries are hashing to random locations as they should. However, if this number is high, data clustering may be taking place. If this is happening the keys should be examined to determine why the clustering is occurring. The effect of clustering is that DBPUT operations may gradually slow down.

A frequent cause of clustering problems is that binary keys are being used which hash to the same location. An example of a poor binary key candidate is a key consisting of a year prefix followed by a sequential number, e.g. 78001, 78002, ..., 79001,79002, 79003, ... If data for multiple years is kept, these key values will cluster. This type of key assignment should be avoided for binary keys.

IMAGE hashing works best when master data set entries are prime or nearly prime. Prime capacities tend to produce hashes which are uniform and not prone to clustering. DBLOADNG flags master data sets whose capacities are not prime.

For master data sets, the chain length information refers to the synonym chain. The maximum chain length shows the worst case number of entries on the synonym chain. Synonym chain lengths should not exceed 1.4 or excessive collisions may be occuring and the hash function should be examined.

DETAIL DATA SETS

The analysis of detail data sets concentrates on capacities and chain characteristics.

Extra capacity is wasteful of disc space and can be easily spotted. Likewise, insufficient spare capacity can be spotted before it develops into a crisis.

Chains which are sorted are marked since additions to long sorted chains can be very slow. Long sorted chains should be avoided if at all possible, or should only be used for data which is relativly static. The reason that additions are slow is because IMAGE must do a sequential traversal of the chain to find where the new record fits in. Not only can this take a long time, but it is also likely to flush the buffer pool.

The primary path for each detail data set is indicated. Unless DBUNLOAD/DBLOAD are used periodically, primary paths are of little value. If DLUNLOAD/DBLOAD are used regularly, then the average chain lengths can be used to determine which path can best be served by making it primary. The most benefit will come from selecting a frequently accessed path whose average path length is long. Paths whose path lengths are short (1 or 2) will receive little value from primary path assignment.

The maximum, average, and standard deviation of chain lengths can be used to get a handle on chain characteristics. Short chains with small deviations and a minimal worst case are best for inquiry when record lookups take place along a chain.

Use of the three chain length values can show when the average case is very good but one degenerate case exists. These lengths are also useful for tuning application cases for the usual or 90% case.

Examination of capacities of multiple data sets may show application integrity problems. Some data sets require corresponding entries in multiple data sets. For example, an employee master data set may be keyed by employee number and an inversion master data set may be keyed by social security number. The numerical counts of these two data sets should agree. If not, the situation must be looked into.

While DBLOADNG can point out several potential problems with a data base, its capabilities are limited. Some ideas on what additional information would be helpful along with conjecture on how to obtain the information are presented below.

Buffer hit ratios to determine the optimum values for buffer pool specifications would be very helpful. IMAGE would have to be modified to provide for buffer statistics collection. DBUTIL could be modified to turn on and report on these statistics. Analysis of hit ratios must take system throughput into account so that buffer hits are not optimized at the expense of system throughput.

A determination of which data sets are most active would also be helpful. The most active sets could then be placed on discs so head movement is minimized. Though, given the difficulty in placing data sets optimally on a disc with MPE, this may have marginal value. As disc drives become larger, the importance of proper disc placemant of files becomes more important. MPE could be enhanced to view a single disc as a set of pseudo discs each of which is sized in manageable units.

CONCLUSION

Optimization of resource usage is an ongoing requirement that requires attention so response time does not degrade. A constant vigil should be maintainted so that no unnecessary requests are placed upon IMAGE. Puts and deletes should be minimized, spurious paths should be eliminated, sorted paths shuld be eyed with caution, capacities should be reviewed regularly, and hashes should be inspected. With the use of DBLOADNG, more information is available to check these problem areas and optimize an IMAGE data base's performance.

REFERENCES

[1] System Performance Measurement and Optimization, Hewlett-Packard General Systems Users Group, 1979, Jim Squires and Ed Splinter.

[2] ADAGER/3000, Rego Software Pty., Calle del Arco 24, Antigua, Guatemala, Telephone: (502-2) 32-4336.

[3] IMAGE Data Base Management System Reference Manual, Hewlett-Packard part number 32215-90003, 9/79.