An Automated Technique for Designing Optimal Performance IMS Data Bases

Vincent Roach
Comress, Inc.

Abstract

There exists great potential for IMS performance improvement through selection among design alternatives. One such area is data base design, which covers a number of sensitive variables. The only feasible way to evaluate the effects of varying combinations of values for these variables, is via simulation. Many of the drawbacks of simulation have been alleviated by the creation of a comprehensive, validated model of IMS, which feeds mostly on pre-existing data. This model allows the evaluation of many alternatives, including data base design. A case study has shown the great potential which may be realized through use of this tool.

* * *

Now that Data Base-oriented information systems have been thoroughly discussed in concept for over a decade, and some actually implemented, there has come to exist some rational basis and motivation for analysis of their performance.

This paper deals with the quantitative aspect of such performance. We address an objective of cost-performance optimization-attaining acceptable levels of system response and throughput with minimal expenditure for hardware. We will be concerned with how we may pursue this objective while engaged in data base design activities. To further restrict the scope of this discussion, a particular data base system, IBM's Information Management System (IMS), will be used to illustrate the techniques and considerations to be presented. We assume that the reader is familiar with the concepts of IMS, as well as the technical jargon used to describe them.

Many aspects of IMS significantly impact performance: system generation, the telecommunications network, design of application programs, specification of the operating environment (OS type, size of regions and buffers, etc.), the manner in which the hardware is configured, to name a few. Based upon studies performed by Comress, a significant determinant of IMS cost/performance which may be isolated for analysis is the design of the data bases. There are a number of design alternatives, the most important of which are as follows:

1. Grouping of fields into segments
2. Placement of segments within the hierarchy
3. Selection of an access method
4. Selection of pointer types
5. Specification of record and block sizes
6. Creation of one or more data set groups
7. Device assignment
8. Allocation of distributed free space
9. Blocking factors for records

We will briefly discuss the key factors involved in each of these areas, then describe a technique allowing the data base designer to optimize his selection among the enormous number of alternatives available.

Grouping of Fields Into Segments

The more fields grouped into a logical segment, the less overhead there will be in the record for pointers, counters, and other system space requirements. However, if fields not logically related to one another are placed in the same segment, extra I/O activity and/or data manipulation may be required each time an access not dealing with all fields is generated.

Placement of Segments Within The Hierarchy

Heavily accessed data should be placed nearest the beginning of a record to avoid unproductive accesses and manipulation of intervening data. Of course, this must be logically consistent; a name of an employee for example, must precede the employee's attributes (e.g., skill, education).

Selection of An Access Method

There are four basic methods of organizing IMS data bases, each with its own access method.

HSAM is strictly sequential, which makes it ideal for sequential batch processing, and horrendous for any random reference.

HISAM lends itself to both sequential and direct access, but: requires reference to as many as three levels of indices before target records may be located, requires accessing of all segments physically prior to the target segment of a record, may generate extensive overflow chains, allows secondary data set groups beginning only at the second hierarchical level and, when segments are deleted, space is not freed until reorganization.

HDAM is subject to none of these problems, but is strictly a random organization and so may be extremely inefficient for sequential processing. Also, some segment chaining may occur due to key synonyms.

HIDAM has an index organized like HISAM, which suffers from the noted HISAM problems. It lends itself to much easier sequential processing than HDAM, since its index is in key sequence. The data portion of HIDAM allows freed space from deletions, and otherwise avoids the HISAM costs.

Depending upon the other characteristics of the data base (record lengths, etc.) and the manner in which it is processed, there may be great variations in throughput, response, and direct access space among the four organizations.

Selection of Pointer Types

Aside from the fact that pointers require additional direct access space (and therefore generate more I/O time when records are accessed) the manner in which segments are connected by pointers can affect throughput significantly.

While a full complement of the various pointer types will allow selection of optimal access paths during processing, this practice costs random access space, and may impact throughput severly when deletions or insertions are processed (all pointers have to be adjusted to account for new or no longer existing links in the chain).

Specification of certain pointer types may lead to the execution of inefficient access paths to target segments. The problem, as in most of the other areas, is that not every procedure which deals with a given segment or data base, does the same type of processing. What is an efficient path for one program's processing intent, may be extremely inefficient for another. Solving this problem means providing the best possible set of results, considering all procedures at once. The technique that is appropriate for this will be discussed below.

Specification of Record and Block Sizes

Physical sizes of records and blocks impact many areas of performance. Obviously, random access space utilization is affected. Short blocks generate low packing densities. Block and logical record sizes not in line with variable record lengths will also generate wastage of space within physical records; additional pointers and additional accessing, increase channel, device, and CPU time. For access methods using overflow areas, improper lengths may cause records to generate excessive overflow chains. Also, variations in block size among data bases will cause excessive overhead in the IMS buffer compaction routines.

Multiple Data Set Group Option

Multiple Data Set Groups can ease file space problems by providing different record and block sizes for segments of varying length. However, there may be additional index accessing required (HISAM), and there will certainly be additional control block processing overhead to determine in which of multiple groups a segment resides. Multiple data set group generation is also an alternative to the creation of multiple data bases. In this case, however, operations addressing the entire data base (sequential processing, reorganization, etc.) must deal with all the data in the combined data base, whereas it would be possible to deal with only the data of interest if it were segregated into a distinctly separate data base. Also, for HDAM, multiple groups beyond the first are all housed in overflow, and so cannot be addressed directly via key randomization.

Device Assignment

Assignment of data base portions to devices affects throughput directly in terms of device timing differences and indirectly in terms of device effects on contention and physical record space.

The larger a device, the more data can be housed on it. Concentrating data on one module limits potential access overlap (hence breeding throughput-limiting device contention). Concentrating data already on one device, into physically adjacent areas, however, reduces access time by limiting seek mechanism movement and head switching. The physical size of tracks and cylinders on devices influences the size of efficient blocks and records, the impact of which has been discussed previously.

Allocation of Distributed Free Space

In HDAM and HIDAM organizations the data base and free space in blocks may be specified at data base generation time. Allocation of large amounts of free space may require large random access space. However, this space may positively influence throughput time. Inserted segments may be placed in blocks where there is free area—these blocks usually being the blocks containing segments from the same record—rather than using new blocks at the end of the data base. In the latter case, retrieval of the entire record would require additional accesses to areas requiring longer seeks. File space is useful in the event that block space is insufficient. New blocks to use may be found closer to the related blocks than if they were placed at the end of the data base. The need is to allocate space sufficient for future use without wasting random access area. This waste is in terms of space and in terms of increased sequential processing time resulting from the spread out data base.

Two other options that bear mentioning fall into this category. These are specification of maximum insert size and segment insert scan limit. In HDAM and HIDAM, the user may specify the maximum space to be allowed for any unique occurrence of a stream of segment inserts. The intent is to prevent one or more outsize segments from wasting block space. The problem is to specify this limit properly to avoid leaving block space unused while most inserts are placed in new blocks. When a new block is required for an insert, the access method begins scanning the random access area for available file space (or another block with sufficient block space) beginning with the block that proved insufficient. The scan limit is set to stop this scan before it reaches the point of diminishing returns. The theory is that when an area of a data base becomes very packed, it will take longer to locate free space than it is worth, and one might just as well go right to the end of the data base to establish residence for a new segment. The trade-off possibilities here are obvious and are a function of device characteristics, free space, block and record sizes, segment size, and a number of other related factors.

Blocking Factors for Records

Blocking factors influence random access space, channel utilization and I/O Control time. The considerations are very similar to those involved with record and block sizes.

Up to this point, the discussion has had a flavor similar to the few notations about performance found in IBM's official IMS documentation. Rare mention of performance-oriented considerations are seldom, if ever, accompanied by any instructions as to how such considerations are to be dealt with in practice. Solutions to this lack of direction typically follow one of two directions. Either no serious consideration is given to performance, or an individual "expert" implements whatever measures seemingly have been successful in his past (and, unfortunately, usually unrelated experience). Obviously, these approaches vary from no effect to partial improvement. What, then, is a reasonable approach? If we examine the set of performance prediction tools generally available, the situation is revealed as follows:

1. Benchmarks are useless, as the system being addressed is far too complex, and more often than not is in the design state so it does not allow testing of any actual data bases and programs to exercise them.

2. Analytical solutions do not exist, if in fact they could ever be developed. The system in question is far too complex for reduction to any simple set of formulas.

3. Simulation, as the last remaining entry in the field, holds the most promise. Simulation is subject to a number of constraints and considerations which must be carefully weighed by the performance analyst.

To build a simulation model of a complex system such as IMS requires a significant expenditure of manpower. The research required probably accounts for more of this cost than the actual programming and checkout of the model.

Validation of the model is a difficult and time-consuming task, full discussion of which is beyond the scope of this paper; suffice it to say that this is a key activity in simulation, and requires great care, control, and time. Finally, as an experimental method, simulation requires: effective experimental design, perceptive analysis of results, and rational-results based induction leading to feasible recommendations. Of course, the model should be built with sufficient flexibility and capability to produce comprehensive and coherent output, to facilitate its proper use.

We have performed the necessary research and constructed and validated a flexible and comprehensive simulation model of IMS. This model, fed by simple parameters describing data base/program designs, has been used by Bell Telephone Laboratories, Pratt Whitney Aircraft, and General Motors, to design, optimize, and predict

hardware requirements for a variety of IMS applications.

This technique, named "SCIMS," allows experimentation with alternative program designs, hardware configurations, system parameters, workload scheduling, and contention with non-IMS jobs, as well as data base design. With the exception of detailed processing descriptions (one statement per DL-1 call is required), virtually all input required by SCIMS consists of IMS control block generation input data (DBDGENS, PSBGENS, etc.) unchanged from the form required by IMS. This means that not only is the analyst freed from model building, but for the most part from the description of the system he wishes to evaluate.

SCIMS runs on any configuration which will support IMS, requiring from 10-60 minutes per run. Each run produces a set of reports which allow all the analysis required to evaluate alternatives in each of the areas mentioned previously. Briefly, these reports include the following:

1. Data Base Description, showing random access module residence, space requirements, logical and weighted average record and block lengths of all data base physical portions (e.g., HIDAM Index Master, Cylinder and Track Indices, HIDAM Index Prime and OSAM; and HIDAM Prime Data for HIDAM Data Bases). This report also lays out the logical structure of a data base by segment, showing name, length (including pointers and other overheads), displacement, probability of occurrence, and targets of every pointer type. This report reflects the data bases as of the beginning of a SCIMS simulation.

2. Environment, showing type of operating system, number and sizes of processing regions, and IMS buffer sizes.

3. Lockout Conditions listing every situation generating a transaction scheduling delay due to segment, data base, program, or occurrence lockout constraints.

4. Data Base Activity Summary, tallying for each segment, each data base, and the entire system, the number of reads and updates generated by the defined processing workload.

5. Transaction Summary, tallying the number of occurrences of each defined transaction by time period.

6. Transaction Description, showing for each transaction the name, priority, calculated PSB size, and frequency of occurrence. This report lists the sequence and volume of all DL-1 calls issued by the transaction, along with the I/O activity generated therefrom.

7. Data Base Description (post processing). This report is identical to #1, except that values reflect the status of the data bases after occurrence of the defined processing workload. This report will be identical with the first

report out of the next SCIMS run, barring prior data base reorganization.

8. Application Accounting. Primarily for verification of input, this report looks exactly like the report of the same name produced from the IMS Log for a live system.

9. Message Processing Statistics. This report, formatted like the IMS-produced version, shows average, best, and worst response time for each transaction.

10. Detail Transaction Analysis. This report breaks down all I/O and processing for a single occurrence of a transaction, reporting on device, channel, memory, and processor times.

11. Detail Schedule. The Detail Schedule produces a report line for every initiation or termination of a control or processing region transaction, reporting time, transactions in process, memory in use, devices in use, and utilizations of channels and processors.

12. Data Base Index. This report cross-references data bases to transactions using them.

13. A number of other utilization reports, showing configuration utilization and resultant cost by component.

Each of these reports is optional.

SCIMS consists of four functional phases:

Phase one is a standard IBM BAL Assembly, using a special macro library with substitute definitions for standard IMS generation macros. The output of this run is a stream of data which is processed (by a temporary program assembled on-the-fly) to create or update a set of AMIGOS (COMPRESS ISAM replacement) work files which contain data descriptive of the IMS system, data bases, and transactions.

Phase two, a highly modular, extremely complex program, processes work file records against one another to simulate DL-1 Activity.

Phase three executes actual time driven simulation using input produced by the previous phase.

Phase four processes output data from the prior two phases to produce the output reports.

When all this is completed, one of the work files contains data representing the status of data bases after processing the defined workload. This file may be input to the next run, to allow SCIMS to simulate the effects of processing overtime, as records come and go, segments are added/deleted, overflow chains build, etc.

Approximately 30 man years of effort have been devoted to this system, which is now in the final states of preparation for official commercial release. It is maintained on a current basis, now being revised for IMS/VS.

To illustrate the value of such a tool in the area of data base design, below are "before" and "after" figures for an IMS system whose design was described as "complete" by the user prior to SCIMS analysis. Only relevant attributes of the system are presented. The figures represent values given an identical transaction workload over a 5-day week of 10-hour IMS processing days. Processing was mostly message with some batch.

Attribute	Before	After
3330 Modules	40	28
2301 Modules	1	2
Memory Utilized	560K	520K
Processor Utilization	68%	32%
Channel Utilization	17%	11%
Avg. System Response Time	2.7 sec.	.6 sec.

The optimization required approximately 5 elapsed weeks. Initial IMSGEN, DBDGEN, PSBGEN data were available. The results show not only a decrease in hardware costs, but a dramatic improvement in system response time for message processing. The procedure consisted of:

1. Running SCIMS for the given design parameters.

2. Analyzing the data to determine sensitive variables over the entire system.

3. Specifying extremes for all variables to determine limits of potential gain in terms of utilization and response (e.g. all drums for data base residency).

4. Reducing extreme values which involved costs, until costs were reduced to acceptable levels (e.g., only one additional 2301, 520K memory instead of 720K) without a large decrease in performance.

5. Fine tuning via variation of sensitive variables in specific areas (e.g. placing a HIDAM Master Index on 2301 drum for one data base, creating a secondary data set group for a particular HISAM data base, increasing a data base blocking factor by 4 bytes, etc.