MAINFRAME JOURNAL

F… of IBM …/370 Architecture & Compatible Systems

December 1989

Should You Tune Your System?

UNLEASH THE POTENTIAL OF DB2 AND SQL/DS.

A full set of database tools to enhance performance and simplify administration.

Making the most of your DBMS is a lot easier with the right tools. Now there's a full set available from Systems Center for two of today's most popular relational environments: DB2 and SQL/DS.

Our DB2 software products (DB/SECURE,™ DB/AUDITOR,™ DB/REPORTER,™ and DB/OPTIMIZER™) address urgent needs with innovative, effective solutions — streamlining security, simplifying auditing, speeding report generation, and boosting system performance. All while eliminating the errors and delays associated with manual DB2 administration.

In the world of SQL/DS, our DB/REORGANIZER™ product dramatically enhances performance and efficiency in a fast-changing database environment. Our DB/EDITOR™ offers exceptionally convenient full-function table editing. And our DB/REPORTER,™ with its outstanding data manipulation and formatting facilities, makes even complex reports a relatively simple matter.

So why wait? Start making the most of your environment — and yourself. Call or write today: Systems Center, Inc., 1800 Alexander Bell Drive, Reston, Virginia 22091.

800-359-5559 703-264-8000

SYSTEMS CENTER

A NEW YORK STOCK EXCHANGE COMPANY

RELATIONAL DATABASE PRODUCTS VM SOFTWARE PRODUCTS NETWORK DATAMOVER PRODUCTS NETWORK ADMINISTRATION PRODUCTS

CIRCLE #20 on Reader Service Card ▲

6-MFJ-8912

C O N T E N T S

December 1989 • Volume IV • Number 12

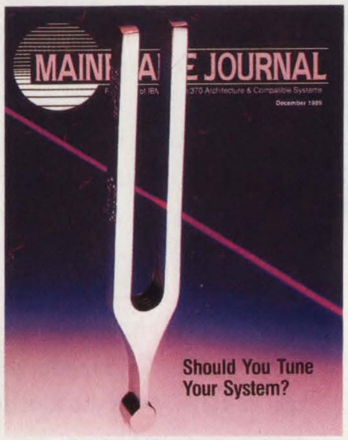
DEPARTMENTS

6 Publisher

8 Reader Forum

10 News Briefs

98 Viewpoint



COVER:

System tuning, computer performance and capacity planning are the Computer Measurement Group's (CMG) primary areas of interest. CMG meets this month in Reno, NV. These topics are also highlighted in this month's issue.

MAINFRAME JOURNAL© (ISSN 0895-5751) is published monthly by Thomas Publications, Inc., 10935 Estate Lane, Suite 375, Dallas, TX 75238, (214) 343-3717, FAX (214) 553-5603. Second class postage paid at Dallas, TX. SUBSCRIPTION RATE: Subscriptions are free within the USA and Canada. One-year foreign subscriptions are $96. POSTMASTER: Send address changes to: *MAINFRAME JOURNAL*, P.O. Box 551628, Dallas, TX 75355-1628.

MANAGEMENT

12 **Should You Tune?**
In a poorly-tuned system, there will be many waits for I/O service. ***By R. Douglas Swords***

20 **Expert Systems For CPE: On Evolution And Data Analysis**
Build comprehensive and useful expert systems for computer performance evaluation. ***By Bernard Domanski, Ph.D., and Sid W. Soberman***

66 **Eight Common Capacity Planning Mistakes And How To Avoid Them**
Avoid these mistakes for a top-notch presentation. ***By Mike Stackpoole***

75 **Artificial Intelligence And Natural Expert Systems**
The problem in general acceptance of AI is the discrepancy between practical achievement and future promises. ***By Roland Braun***

96 **A Perspective On Performance Management**
DP management's focal point should be the company's goals. ***By Reba L. Chaisson***

98 **Viewpoint: The Illusion Of Measurement**
It is easy to confuse accessibility of measurement data you enjoy with the availability of data that is needed. ***By H. Pat Artis***

SYSTEMS

24 **VSAM Tuning For Data Entry**
Dataset tuning is an iterative process that requires making choices. ***By David Shein***

32 **MVS Program Management**
Program management for MVS performs three functions: searching, fetching and synchronizing. ***By Robert H. Johnson***

44 **Overcoming CPU Constraint In Large CICS Systems**
CPU constraints on CICS can be relieved. ***By Ted C. Keller***

56 **VM Data Communications**
VM is best known for its interactive timesharing support. ***By Ed Sterling***

70 **VSE/VTAM Interfaces**
VSE users will gain an understanding of the purpose, terminology and basic knowledge of VTAM. ***By Leo Langevin***

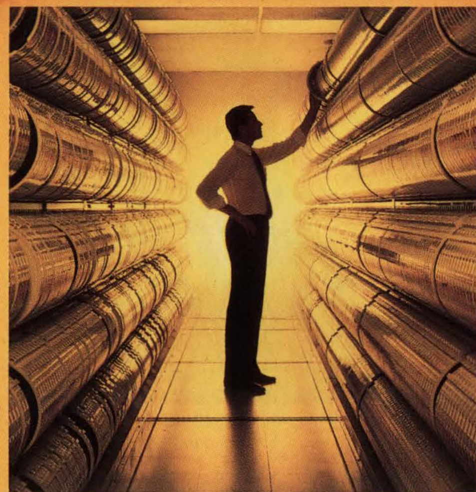
APPLICATIONS

62 **The ISPF Productivity Update**
Readers' ideas about program function keys and ISPF edit macros are presented. ***By Jon E. Pearkins***

79 **SQL/DS Database Reorganizations**
Review operational and technical issues associated with implementing a database reorganization system. ***By Michael Teitelbaum***

84 **Referential Integrity In DB2**
You will come out ahead using DB2's referential integrity capability. ***By Lawrence Stevens***

90 **Debugging With COBTEST**
Familiarize yourself with the use and benefits of interactive debugging. ***By Alan Friend***



Computer Measurement Group (CMG)

Bob Thomas

Wringing the very last drop of performance out of existing mainframe systems is the goal of most MIS Directors and Technical Support Managers. For the past 15 years, the Computer Measurement Group (CMG) has worked to provide DP professionals with the latest developments related to the performance, planning, management of computer systems and the use of Computer Performance Evaluation (CPE) methodologies. CMG's annual conference this year will be held December 11-15 in Reno, NV.

Although not specifically oriented to users of IBM mainframe computer systems, the reality of the marketplace dictates that the vast majority of CMG's attention is focused toward users with IBM and compatible mainframe systems.

CMG's primary stated objectives are to provide:

- An extensive introductory educational program for the development of new CPE professionals
- Continuing education for existing CPE professionals
- A forum for the exchange of information, promotion of new ideas and discussion of management information requirements
- CPE professionals with focus on practical applications and results-oriented methodologies
- Encouragement for educational institutions to focus on CPE curriculum.

Not only will *MAINFRAME JOURNAL* be represented at this years CMG (booth #26), but also this month's issue contains several articles oriented specifically to capacity planning, system tuning and performance issues.

If you are interested in knowing more about CMG, give them a call in Chicago at (312) 938-1228.

Mail Room Bandits

During the past six months we have received several phone calls from subscribers wondering why they did not receive their issue of *MAINFRAME JOURNAL*. Almost without fail, we had their correct address and, in fact, their magazine was mailed out. After tracking down a number of these instances, our conclusion is that some company mail rooms are taking it upon themselves to decide which mail gets delivered and which mail gets trashed.

Even more bizarre, we have received several letters from mail room personnel cancelling the subscriptions of MIS personnel. When we contacted several of these "cancellees" to inform them, they were shocked and irate. They were especially upset that someone in their own organization had the gaul to arbitrarily cancel their subscription to a publication designed to enhance their abilities as well as the performance of the organization's data processing capabilities without their consent or notification.

A recent article on this same subject in the Wall Street Journal (October 26, 1989) confirmed the suspicion that this outrageous practice is becoming more widespread. According to the article, several large, well–known companies are simply not delivering magazines even when addressed specifically to an individual. In the article, Michael Bronner of Bronner Slosberg Associates, a Boston direct-mail firm said, "It smacks of big brotherism. They're going to decide what their employees can or cannot read."

If you suspect that your mail room is taking it upon themselves to decide what you read, let us know and we will be happy to send your magazine to your home address.

Bob Thomas

MAINFRAME JOURNAL
For Users of IBM System 370 Architecture & Compatible Systems

Publisher and Editor-in-Chief
Bob Thomas

Associate Publisher
Martha Thomas

Associate Publisher Corporate Services
Suzanna Penn

Editor
Carol M. Hoag

Copy Editors
Judy Beller
Cecilia Perry
Pat Warner

Art Director
David Kramer

Assistant Art Director
Ken Buerer

Production Manager
Diane Dishman

Marketing Services
Sally Webb

Circulation Manager
Janice Porter

Assistant Circulation Manager
Nancy Crawford

Administrative Manager
Marian Davenport

Advertising Sales
Denise Thomas - Mark Cauto

BPA

List Rental Infomation
Contact AllMedia (214) 9854060

SUBSCRIPTION RATES & INQUIRIES: Subscriptions are free within the USA and Canada. One-year foreign subscriptions are $96.

Allsubscriptions, remittances, requests and changes of address should be sent to *MAINFRAME JOURNAL* at 10935 Estate Lane, Suite 375, Dallas, TX 75238, (214) 343-3717.

W. C. Fields & La Dolce Vita

I imagine that there are many EDP professionals who can readily identify with the "wizard" in Thomas E. Williams' Reader Forum letter in the October 1989 issue. Every new frontier in the history of mankind has been braved by the pioneers of their age: men and women who risked it all in the pursuit of adventure, discovery, and, of course, riches beyond measure. Future generations will study our Computer Age with detached fascination and school children will snicker at the thought of computers that ran on electricity rather than light. Grandma will browse the family photo album and mumble something about her dad who had been "in on the ground floor of the Computer Age" and who, for reasons obscured by time, missed his golden opportunity to strike it rich, settling, instead, for a gold-plated corporate handshake and car fare home.

I suppose we should be grateful, though, for La Dolce Vita that EDP has afforded most of us and our families through the years. For strange and introverted personalities there are few other professions which could have offered a comparable chance at a normal, middle-class lifestyle. Of course, the cost of this success has been great for many of us - stress induced drug dependency, burnout and the dreaded chronic vegetative state syndrome are always lurking just around the corner. It is time for the new generation of "wizards" (now called computer scientists) to come forward and, as W.C. Fields once said, "Take the bull by the tail and face the situation."

Tom Dobbins
New Haven, CT

Selecting An Index Control Interval Size

In reference to the article "Selecting an Index Control Interval Size" by Michael Sachais in the November 1989 issue, the writer suggests moving a CICS file from LSR to NSR if the INDEX CISIZE exceeds the maximum allowed for the installation's INDEX pools. It should be noted that an alternative is to reduce the Control Area Size (CASIZE). This would result in fewer CIs per CA, thereby reducing the number of keys required to fit in the index record. The CASIZE is reduced by specifying a secondary quantity less than cylinder size. This strategy has been verified at DFP level 2.3.

Larry Bridges
Indianapolis, IN

Clever Savings

Kenneth McBride's DMKSNT article in the November 1989 issue was quite interesting and showed a clever use of saved systems that I intend to make use of.

I would like to add a few suggestions that will make the creation of these saved systems a bit easier. XEDIT the stand-alone program you want to save (IPL FMT S for his example). Go to the last line and HEXType it. The 6th, 7th and 8th bytes will show the entry point address of the program, X'000600' "PER Instruct Range 600", which will set a PER stop at address X'600'. Punch the file to your reader (don't forget the "(NOH") and IPL the reader. When you reach the PER, stop a few seconds later, then do the SAVESYS. If you do use ADSTOP,it must be done as McBride describes because ADSTOP stores a trap instruction at the location of the stop and this trip will be overlaid by the loaded program. PER stops are set in control registers and do not overlay storage. ADSTOP is left over from CP-67 days, which is before PER was invented and still exists for upward compatibility.

One further hint: If you try to save DDR in the same way, you will need at least two segments as DDR is larger than 64K. I have not tried this on VM/XA/SP but see no reason why it would not work there. Just use a DEFSEG command instead of the DMKSNT entry and make sure that you are in the proper machine mode. DDR and FMT would need a 370 mode machine, while DDRXA would need an XA mode machine. Since VM/XA/SP segments are one megabyte instead of 64K, one should suffice for any of the s/a utilities.

Rich Greenberg
Los Angeles, CA

Program Exceptions: Another Solution

I agree with most of what was said in the "Understanding Program Exceptions" article by Harvey Bookman in your October 1989 issue. Most programmers do not understand the causes behind program exceptions and ABENDs. But I don't think educating them in what causes a particular program exception of ABEND is a practical approach to the situation. Nor is training in the proper use of manuals an effective approach. Most installations would have a tough time cost justifying training their staff in ABEND and program exception handling, especially since training dollars could be put to more effective use by covering other topics. Also, the cost of purchasing message manuals in volume and trying to keep them up to date would be prohibitive.

A more productive approach to the situation would be to use the machine to provide the solution. Software packages are available that proces the dump for the programmer, providing not only information on what caused the exception/abend, but also identifying the statement and data causing the problem. In most cases a programmer would never have to look at pages in a dump. An on-line copy of the messages manual would eliminate the problems of cost and maintenance associated with using a hard copy manual system to provide the information. Let the machine provide the information required to resolve the problem, it's the productive approach to the situation.

Jim Feurig
Rochester Hills, MI

ISPF Tips

As an experienced ISPF/PDF user, I find the text processing commands to be very helpful in performing many host-based word processing functions. I'd like to add to the tips for using ISPF/PDF's text processing commands in Jon Pearkin's article, "ISPF And Text" (September 1989).

For Text Split (TS) to be useful, I find that assigning TS to a PF key is essential. This enables one to place the cursor at the position in the text that is to be split and then press the assigned PF key to perform the split. Likewise, I assign Text Flow (TF) to a PF key. The combination of TS and TF via PF keys enables me to quickly insert a word, phase, sentence and so on in the middle of text and quickly flow the paragraph back into shape.

David Shein provided information on assigning EDIT LINE commands to PF keys in his article, "Raising Your IQ (ISPF Quotient)" (October 1989). An example of assigning TS and TF to PF keys follows:

```
Use ISPF 0.3 or enter KEYS on the command line
PF1 === HELP
PF2 === SPLIT
PF3 === END
PF4 === :ts    /* PF key 4 will now split a line */
PF5 === :tf    /* PF key 5 will now flow a paragraph */
```

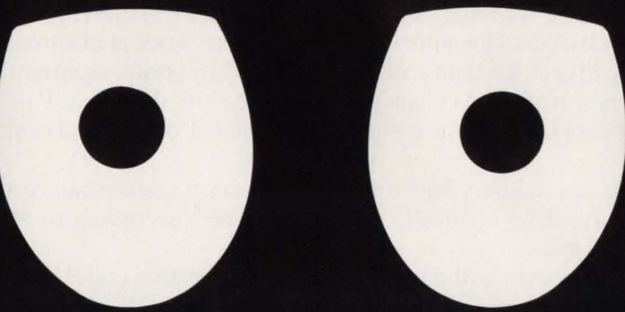
John D. Hillmer
Milwaukee, WI

Mainframe Spreadsheets

I appreciated the article "Mainframe Spreadsheets Take Off Again" by John Kador in the November 1989 issue, however I feel the article may have been misleading to many of your readers. Although Lotus announced its mainframe spreadsheet in April of 1987, their product is still not available for purchase (as of Nov. 9, 1989). The article also suggests that many of the "innovative" features in the Lotus product had previously been unavailable. This is incorrect as ESS (our mainframe spreadsheet) has been available since 1982; has had 3-D capability since 1985 and direct DB2/SQL access since 1988.

We have watched *MAINFRAME JOURNAL* grow and mature from its first issue to its present position as one of the most complete and informative magazines available for MIS managers and users. However, this article serves to perpetuate the "success-by-marketing" concept so prevalent in the mainframe industry.

F. Thomas Cox
Trax Softworks
Los Angeles, CA



This is one place you'll never be when you buy our software.

When you choose any DTA software product there are many things you'll get. Like very effective, easy to work with, low-frustration products.

There is, however, one thing you won't get. You won't get left in the dark. That's because we support our software products with clear, exacting, thorough documentation. Documentation that many consider the best in the industry.

And there's more. You also get the DTA Straight Line, a direct customer service number that puts you person to person with DTA's Software Product Developers. So in the unlikely event that you hit a snag, you can get an answer to your dilemma straight from the horse's mouth.

Our software product developers work day and night fine tuning our products. Every detail is painstakingly covered. And all new products are tested under tough, honest, fool-proof conditions at client sites.

So if you're looking for some select software products to improve your VSE, MVS or VM systems, give DTA a look. *DTA software...More efficient in every detail.*

- ☐ DTA/RECOV–CICS VSAM Forward/Backout File Recovery.
- ☐ DTA/PRINT–Printing and viewing facility for POWER and VM queues.
- ☐ DTA/QCOPY– Queue management for POWER and VM queues.
- ☐ MENU–VTAM Session Manager Menus, Windows, Electronic Mail.

DTA®

Software Products Group
550 Waterford Park
505 North Highway 169
Minneapolis, MN 55441
800-521-6773
612-591-6100

CIRCLE #129 on Reader Service Card ▲

IBM Streamlines Batch Processing And Offers System-Wide Security

The new release of MVS/ESA recently announced by IBM provides high-speed batch processing, enhanced systems security and improved operating system performance.

Hiperbatch

Hiperbatch, a new function of MVS/SP 3.1.3, represents the latest integrated hardware/software enhancement providing up to a 60 percent reduction in the time necessary for processing batch. Hiperbatch uses ES/3090 processor memory to store data, reducing the need to repeatedly access disk storage devices. This allows the processor to work at electronic speed instead of the slower speed of mechanical devices to retrieve data. Hiperbatch jobs can access the same information concurrently, speeding the processsing of multiple applications. Hiperbatch relies on the new hardware enhancement to ESA/370, the Move-Page Facility, to speed batch processing. It is a highly efficient method to transfer data between the computer's expanded storage and central storage. The Base Control Program will be available December 22, 1989.

"This is the tip of the iceberg in a strategy to marry hardware and software for performance at the end-user level. Hiperbatch allows selected VSAM or QSAM datasets to reside dynamically in a Hiperspace," according to Marty Clague, IBM's Assistant General Manager of Marketing for Enterprise Systems.

Hiperspace is created by the operating system without changing any application code. Furthermore, the dataset can be owned by a single task, shared among job steps or shared among jobs. "The user can load the data, share it and provide all the necessary security controls," Clague maintains.

Security

"IBM has taken a systematic approach to security by putting all the functions under a centralized point of control in MVS/ESA using RACF," Clague points out. The Resource Access Control Facility (RACF) Version 1 Release 9 is IBM's strategic program for providing security support for MVS/ESA and MVS/XA, VM/XA SP and VM/SP with or without the VM/SP HPO. In conjunction with other designated products, RACF 1.9 will meet U.S. Department of Defense criteria for a trusted system at a B1 security level.

RACF functions that protect computerized information include security classification support, system operator control, job submission control, network security support, surrogate user support and VM shared file system support. In MVS/ESA, it includes support for the new Hiperbatch function.

RACF 1.9 will be generally available for MVS and for VM/SP in September and for VM/XA in November, 1990. In addition Transaction Security Sytems products utilize cryptography, smart cards and signature verification to protect network transmissions. Another security option allows customers to isolate processing on any PR/SM partition on ES/3090s. Also, a new range of security services will aid in reviewing and evaluating security policies and develop plans and implement programs to reduce security exposures.

Because of the growing concern with computer viruses, IBM has established a new research project called the High Integrity Computing Laboratory where studies are underway to develop better ways to combat computer viruses or unauthorized code.

Storage

A new 16-model series from the ES/3090 family was also announced. ES/3090 J and JH Models provide up to 4.5 gigabytes (4.5 billion characters) of processor storage, almost doubling the processsor storage available on previous ES/3090s.

The new ES/3090 J Models (180J and larger) provide 7 to 14 percent more throughput in an MVS environment than ES/3090 S Models. IBM's four-million-bit memory chip will be available in select ES/3090 J Models. These new models feature asymmetric central storage to allow logical placement of processing resources on a most-needed basis as opposed to symmetrical placement. They also provide an enhanced PR/SM with resource capping so that processor resources available to each PR/SM partition can be defined to a fixed maximum. Dynamic storage reconfiguration allows central and expanded storage resources to be moved to a logical partition to meet shifting workload demands.

Available now are 13 of the 16 ES/3090 models along with the ability to run VM/XA SP2 in CMS intensive environments in a logical partition with production level performance. Asymmetric central storage is also currently available.

Using 50 percent less floorspace than a 3480, the 3490 Magnetic Tape Subsystem also offers performance advantages including improvement in productivity for tape operators and reduction in system wait time.

Through an improved data recording capability, the 3490 can store up to five times as much data on a single tape cartridge as a 3480 without the capability. It can increase the performance of the subsystem up to 70 percent. All models of the new 3490 family use the same cartridge as the 3480, allowing data to be exchanged between systems.

Printer

A high-speed addition to the 6262 line of impact printers, the Model 22 prints at 2,200 lines per minute, 57 percent faster than current models. A 6262 bar-code feature allows labels or documents with bar codes to be printed along with other information. Both the printer and the bar code are now available.

Should You Tune?

By R. Douglas Swords

The question that often comes up with management during discussions about computer performance, or perhaps more specifically, a formal computer performance management program, is — *Why tune a computer system at all?* It is an interesting question. Its basis probably lies in the belief that because of the current state of technology (solid state disk, cache controllers and sophisticated operating systems) tuning a computer system is unwarranted and perhaps unjustifiable. Is this really true? Could it be due to a misunderstanding as to what the objectives of a performance management program are? You do not always tune to improve response time for users, although that is often a positive side effect. You tune to increase the throughput of a system so that more work can be done *without degrading response time* for the user.

There is a significant audience for performance-related issues. Most larger organizations with complex machine environments have established staff for the purpose of studying and improving the performance of those systems. There are many companies whose products address computer performance. It would seem to follow that there must be sufficient proof somewhere that performance management is indeed justified and might provide benefits in a computer installation of any size.

The I/O Subsystem

Much has been written and many studies done on the performance of central processors, most likely because the power of the processor is how the machine itself is rated. Whether you use Million Instructions Per Second (MIPS) or Internal Throughput Ratio (ITR) ratings, many studies have established the relative power of different classes of processors. There are a multitude of products and tools for evaluating the performance of the CPU. It seems, however, that somewhere in this process the most important part of the picture, the I/O subsystem, has been slighted in attention.

Tune to increase throughput so that more work can be done without degrading response time for the user.

Is the I/O subsystem the most important part of the system? Well, it is true that you cannot do work without a central processor, but from the standpoint of performance the I/O subsystem contributes more to good or bad performance than does the processor. The processor is one entity with one major function. The I/O subsystem will, in most cases, be a complex arrangement of storage devices and controllers that are relied upon to feed the central processor. Since each node in the I/O complex is a point of contention and the majority of service that each entity receives in business applications is I/O, it seems reasonable that this portion of the system is crucial to both successful completion of work and the performance of that work. This would not be true in an environment where the majority of work being done is scientific, the implication here being that scientific applications are primarily number crunchers. Although this is certainly not always the case, if you are trying to find the next largest prime number or executing weather models, you would not expect to do a lot of I/O.

When a central processor is stalled or waiting for some service to complete, it is most likely an I/O request either to retrieve application data or a required frame that has been paged out. In both cases the responsiveness of the requests will depend on the amount of work present, the nature of the attached devices (whether they are cached or perhaps solid state disks and whether or not there is expanded storage available where the requested data or frame might reside) and the performance condition of the I/O configuration.

It is also true that the developers of current technology databases, built on relational models, strive to reduce or improve the I/O process that these data managers use. IBM's offering of MVS/ESA and improved versions of DB2 are good examples of attempts to improve the I/O process.

With all of this considered, how well an environment is configured and how much effort is expended to understand I/O traffic patterns and make necessary adjustments will still have a substantial and measurable impact on system performance.

The Effects Of Slow I/O Response

Do not even consider user perceived response time. Look at what happens on an imaginary computer system when response from the I/O subsystem is poor.

A user initiates a transaction request. The system receives this request and must first establish an entry in its internal management tables for this transaction. The transaction goes to the end of the current chain and *waits* its turn. In a single processor asynchronous action takes place so that until the other transactions of equal priority have completed your request *waits*. When it becomes your turn, the system will look for the program you want to execute. If it is not currently resident or residing on a paged out-frame, it must go out to the load library and get it. So you schedule your I/O request. The I/O scheduler gets the request but there are several ahead of you and you *wait* your turn again. As your number comes up, you zip out but find yourself in another

queue because the volume you need is reserved by another system or user. So you *wait* again.

When you finally get to the device, you have to go to the directory to find the location of your program and then go get the program itself. But you are delayed again because the volume is fragmented and the dataset exists in several extents. When you finally get back to the processor, you find that because you took so long your frame was needed and it got paged out. So then your transaction processor had marked you suspended and now you have to *wait* to get reactivated and then *wait* to get your frame back.

You get your frame back and the first thing you need is your map or screen and you are going out again. To make a long story short, you continuously return to the I/O subsystem for service. In a poorly-tuned system there will be many waits for I/O service. The fewer the waits the more throughput is possible.

Hardware Performance Boosters

In today's marketplace there are many options for the I/O subsystem. The standard two-path control unit has been replaced by the four-path unit and the four-path will probably soon be replaced by the eight-path and so on. As the storage devices become denser, more paths become necessary to reduce the impact of path delays.

Cached control units are designed to eliminate as much I/O to a mechanical device (DASD) as possible. When the data needed is found in cache, you are exempted from such delays as RPS delay, seek, latency and IOS queue. The resulting I/O may be only four or five milliseconds or less where it could be 20 to 25 milliseconds if it had to be serviced by DASD. Newer cache offerings include the ability to cache or buffer updates. The read-to-write ratio that was important for read-only caches gives way to considerations based only on the access patterns and rates relative to other datasets sharing the same cache.

Solid state devices offer the capability to configure many paths to extremely high-speed storage. They will typically service an I/O request in about one millisecond without the previously-mentioned delays of rotating DASD. They can be used for paging migration or user data and they are substantially more expensive than traditional DASD.

Expanded storage is also an I/O device although it is an extension of real storage. Depending on how you use it as a page migration device determines how much benefit you can receive from it. Some of IBM's software is being optimized under the MVS/ESA operating system to use expanded storage. There are techniques you can use today to get ESA performance from XA if you choose to do so.

What Are The Performance Issues?

There are mountains of information available on tuning the various aspects of the I/O subsystem. I will review some of the major ones.

System Managed Storage

There is a lot of excitement about the possible implications of System Managed Storage (SMS). There has been some change in the way a dataset is allocated when a volume is under SMS control.

Your system should be well tuned before you attempt to migrate to SMS.

For example, if, when defining the Storage Class, a 10-millisecond response time is given as the target, several things happen at allocation time. First, when the Automated Class Selection (ACS) routines intercept the allocation request, a list of up to 15 eligible Storage Groups will be built. Each Storage Group will be scanned to see if there is a volume delivering the requested response. If not, the groups will be scanned again to see if there is a volume that is close. If not, the groups are scanned again and the first volume that can provide adequate space is chosen. But if the requested space must be guaranteed, all the above is not done and the first volume with appropriate space is selected. In addition, the dataset will be cached if cache is present. This is different in that previously, under IOS control, the volume least busy was chosen.

There are several potential problems in the above scenario. First and foremost, all datasets should not be cached. SMS does provide an exclusion parameter. You specify response for the dataset or class as 999 milliseconds and that will prevent caching. It also means that literally any volume could be chosen. There is obviously no performance benefit here.

Next, if you routinely delete VSAM datasets and then reload them to reorganize them, you can run into a problem. If the volume where the dataset resides has been processed by IDCAMS to unload the dataset, then possibly, due to the response time of the volume during unloading, that volume will not meet the performance criteria set and your VSAM dataset would go someplace else. Where? Well, if it is an on-line dataset used by CICS, you are going to be reorganizing the dataset during off hours. There may be volumes in Storage Groups that give required performance at night due to their access patterns, but during peak daytime hours their performance is bad. Or there may be some other dataset not accessed at night, but it is during the daytime and your VSAM dataset could be placed there and then create an IOS queue problem during the day. SMS documentation clearly states that placement of important datasets is a manual function to be accomplished by personnel in the Storage Administration Group. You just cannot get away from it.

A large system under SMS control will have overhead associated with ACS processing itself so that the utilization of ACS, the way the macros are constructed, the number of volumes placed in a Storage Group and the number of eligible Storage Groups become performance management issues. Along with this, if the Data Facility Hierarchical Storage Manager (DFHSM) is being used, the migration of datasets becomes a performance issue. Are the migration timetables properly constructed? Do you use level one (compression on active volume) or go directly to tape? More performance management issues.

Your system should be well tuned before you attempt to migrate to SMS.

DASD

At the volume level you are concerned about seek time, latency and device busy which will contribute to IOS queue. Volume fragmentation will contribute to all

of these conditions. Dataset placement will contribute to seek and a lot of busy little datasets on high density devices will contribute to device busy, seek and IOS queue. Connect time is an issue and can be modified by the blocksize (physical record size) and the number of buffers available to certain access methods. For example, when using Queued Sequential Access Method (QSAM), if a large number of buffers is assigned to a dataset with a large block-size, when the device connects it will transfer enough blocks to fill all available buffers. This is a good way to improve the elapsed time of a sequential application, but it is also a way to build queues and RPS delay.

RPS delay occurs when a path is busy transferring data and somebody else needs to use it but cannot. You usually view RPS delay as an indication of the condition of an entire string, but sometimes RPS delay can be traced to one device on the string and possibly even a single dataset.

New DASD developments, such as the 3380K and compatible devices, create a problem in and of themselves. While the 3380K is three times as dense as a 3380D, it is not three times as fast. So, you cannot place three times the demand on them. What you use them for is space, not performance. So you take active datasets from *one* 3380D and then inactive datasets from others and you have it just about right. Determining the performance capacity in your environment is another example of the performance management function.

Cache

Cache will certainly do one of two things: either improve the performance of applications with access to it or degrade them to the point it is unbearable. The secret lies in the amount of time the control unit spends staging data into the cache. Even worse, improperly-sized cache can place a control unit in a state of constant staging, thus tying up the control unit paths and the volumes from which data is staged.

Why is this? If a read miss occurs, several actions take place. The data is transferred to the requesting program and loaded into the cache. Then, after a signal is sent indicating the I/O request is complete, the staging takes place moving what is left on the track into the cache hoping to successfully anticipate a future data request. While everybody thinks the I/O is finished, the controller sneaks around and ties up the staging volume until the staging is completed. When this occurs a queue can build and the reason for this queue can be hard to detect. This is because the good things the controller does, when averaged in with the bad, make everything look good. It is about three times as hard to detect if the controller is shared because it might appear that another system is utilizing the device when, in fact, it is the controller itself tying things up.

The 3880-23 control unit was especially susceptible to this staging. The 3990-3 has some improvements, the most obvious being the addition of two paths and Device Level Selection Enhanced (DLSE) which allows four paths to access any volume on a correctly configured set of strings. Internal enhancements will also allow the 3990 to simultaneously manage eight operations (four transfers from cache and four connections to DASD for other reasons). While the effects are diminished, they are not eliminated. Unfortunately, the overwhelming urge with new triple density DASD and 4.5MB-per-second channels is to populate a pair of strings of eight or more actuators with this triple density DASD, possibly increasing demand on the controller three times (in this case the assumption is that someone might fold three 3380Ds performing at capacity into one 3380K or that each K on each string will have absorbed the activity of three Ds each). So you go back to the original problem. Only now it is worse because instead of one or two applications behind the controller, there are now several.

Increasing the size of the cache is not the appropriate action to take either. While it might seem to make sense that increasing cache size increases hits, what will probably happen is you will increase the staging situation and make things worse.

The best way to deal with this is to evaluate the utilization of the cache and reduce staging by decaching those datasets which do not cache well. Commercial software can identify controllers with high staging rates for you (I/O subsystem tuning software). Other improvements in caching technology, such as the FAST WRITE option for the 3990-3, have a tendency to create a different set of problems when older problems are reduced or solved.

Solid State Disk

Solid State Disk (SSD) gives you the technology of real storage in a box that the system believes is a control unit with an attached volume(s). Although expensive, these devices can do miraculous things for performance. They do not experience staging, RPS delay, seek or latency and IOS queue is highly unlikely. For example, EMC Corporation (Hopkinton, MA) produces a device, ORION, that may contain up to 16 independent storage directors. When attached to 4.5MB channels, the aggregate throughput rate is 72MB-per-second. The device itself is capable of 900 I/Os per second, probably more than the average installed system generates.

For a critical application you can achieve a fixed service rate of one millisecond per I/O. This certainly exceeds most requirements, but there are places for it. Tuning this device is less of an issue than deciding what datasets could best use it. As with cache, the use of the device must be determined through performance management or capacity could be wasted.

Expanded Storage

Although not really considered an I/O device by a lot of people, I think it really is. The best use of the expanded storage so far is as a page-migration device. It services a request for a single 4K frame in about 75 microseconds. It will come into significant play with MVS/ESA, but there are ways it can be used now.

For example, 64MB of expanded storage can be of great benefit when VIO is used. Even though you are paging out to expanded storage, you are doing so at great speed and without physical I/O. So, you can then make small temporary datasets VIO, small sort work space VIO and so on getting great performance gains, however indirectly, from expanded storage. Adding or increasing expanded storage can make it possible to increase the use of VIO. This same technique can be used to handle page-outs by increasing region sizes and allocating more buffers during batch processing. Although these buffers will be brought into real storage when referenced, you once again enjoy tremendous page-in times.

Now for the downside. You can do it, but do not overdo it. Using the real storage manager to manage this pseudo (we say pa-sway-doe) I/O is all right to a limit, but there is a limit. Each page migrated consumes a certain number of cycles; some sources say it takes 2,000 machine instructions to move a page into real storage. Now, at 75 microseconds per frame you could theoretically move 13,000 frames in per second (1000000/75), but you cannot load down a processor with that much paging. Thirteen thousand frames multiplied by 2000 instructions per frame gives a requirement for 26 MIPS just to move page frames. Most management would frown on this. As an experiment and to generate proof, you might try telling management you need a 3090-150 or thereabouts to handle your regular work and a 3090-300 to take care of your paging requirements. See if management frowns.

Determining the best mix in using expanded storage like this becomes another performance management function.

It is important to report the progress of performance management to management.

Channels

Some of us will have the option to use faster channels, most commonly a 4.5MB channel over a 3MB channel. This channel speed is only availabe for transfers from cache and from certain 3480 cartridge tape controllers. It looks like a lot and it is better, but I will do some math.

The 3MB channel is 33 percent slower than the 4.5MB channel, so expect a similar increase in connect time when the 4.5MB channel is connected to cache and you have a cache miss. So, on a 4.5MB channel, a 9K block will need a two millisecond transfer time on a cache hit and a three millisecond transfer time on a cache miss. On a cache miss the transfer rate is at 3MB because of the head transfer speed of DASD. So once again you have to be careful how you utilize faster channels and take into consideration cache misses. This is another example of a performance management function.

Management Reporting

It is impossible to overstress the importance of reporting the progress of performance management to management. When decisions are made based on results, those results need to be available. Perhaps the best way to report progress is graphically. Charts depicting before-and-after measures, careful documentation of cause and effect and the budget impacts of performance management activity should be maintained and available.

Improving the performance of a device, string or control unit has financial implications. While upper-level management may not understand or even care about *how* the results were obtained, if you can document an improvement in hardware investment, management will notice that.

Conclusion

Tuning the I/O subsystem will improve throughput, reduce paging, reduce address space overhead for transaction processors and improve the dollar-to-performance ratio of hardware. What is the cost?

Purchasing an I/O performance management tool will cost around $25,000 (higher for multiple processors). There will be some cost allocation of salary and benefits for those people involved in the performance management and some downtime necessary to make adjustments. Considering all that, consider the following reasoning. A 3990-3 minimum configuration costs about $200,000; solid state disk about $1,000 per megabyte, so a 256MB SSD is around $256,000. Expanded storage is approximately $300,000 for the first 64MB and real storage far more expensive. A 10 percent improvement in the performance of any of these devices repays the cost of the commercial product. If there are multiples of these types of devices in the environment, it does not take much improvement to repay the investment of salary and benefits.

After that, it is all gravy.

ABOUT THE AUTHOR

R. Douglas Swords is the storage subsystems product manager for Boole & Babbage, Inc. He has been in data processing for the last nine years.

Expert Systems For CPE On Evolution And Data Analysis

By Bernard Domanski, Ph.D,
and Sid W. Soberman

The purpose of this article is to both review the evolution of Expert Systems (ES) for Computer Performance Evaluation (CPE) and to illustrate circumstances where the *comprehensive* ES needs to incorporate historical data analysis. The concept is simplified if a performance database is used as the primary data source for the ES. The overall objective is to provide insight and to stimulate interest in integrating three technologies (ES, statistical data analysis and database management) to better analyze the problems faced by CPE analysts today.

Evolution Of Expert Systems For CPE

In the article "Expert Systems For CPE: On MetaRules and Knowledge Engineering" *(MAINFRAME JOURNAL,* June 1989), the author describes a small set of basic *primitives* that can be used to describe the structure of a knowledge base for an ES for CPE. This set of primitives is reviewed so you will have a common frame of reference. These primitives are defined as follows:

- *prompt (keyword,symptom text)* associates a symptom observed on a system with a keyword; *prompt (hipage, page-fault rate exceeds threshold)*
- *condition (problem,[keywords])* associates a set of symptoms (via keywords) with a system problem causing the ES to extract the data associated with the symptoms specified before checking whether the problem exists; the *diagnosis* primitive (below) actually checks for the presence/absence of the problem
- *depends (problem,[logical expression], keyword)* causes the ES to determine if a symptom (via keyword) exists but does so only if the *logical expression* (of symptoms) is true; for example, the ES would want to extract data to determine if expanded storage were overutilized only if the processor were in the 3090-class
- *diagnosis (problem,[logical expression], treatment)* causes the ES to determine if the *problem* exists — that is, the problem exists if the *logical expression* of symptoms is true; in addition, this primitive provides a *treatment* for the problem — this is equivalent to an *IF-THEN* rule: IF [logical expression] THEN problem exists and treatment should be applied as a remedy
- *diagmsg (problem, diagnostic text)* associates detailed descriptive text with the problem
- *treatmsg (treatment, treatment text)* associates detailed descriptive text with a treatment for a problem; *treatment* here is associated with a problem via matching*treatment* keywords on the diagnosis and treatmsg primitives
- *helpmsg (keyword, help text)* associates detailed descriptive text with a symptom (via keyword) similar to a data dictionary in that a symptom can be defined along with its data sources and any anomalies that are known about that symptom.

Most of today's commercially available ES shells encompass this set of basic primitives in one form or another. Some of the more robust ES shells have incorporated such features as *fuzzy reasoning*. Fuzzy reasoning allows certainty values (weights) to be associated with rules; for example, if you ask whether I/O response time exceeds a threshold, a weight such as 0.5 could be associated with this symptom. You might also associate a certainty threshold to a rule's conclusion; below this threshold, rules are considered to be false. You could extend your set of primitives to encompass fuzzy logic; the prompt primitive could be redefined to include a weight with each symptom. The diagnosis primitive could be extended to include a certainty threshold. A whole set of underlying mathematics exists for evaluating the weights associated with fuzzy expressions. In general, *or*-type operations imply adding the weights and *and*-type operations imply multiplying the weights.

Frame based ESes are beginning to emerge. A frame is like an object with a set of attributes or *slots;* a frame for a microcomputer might have slots for the type of CPU, monitor, keyboard, I/O ports and printer. Frames can be arranged to form hierarchies where lower level frames inherit the characteristics of their ancestors; that is, if a tree has limbs and a pine is a tree, then a pine has limbs. But with frame-based systems, you are released from associating a single type of value with a slot. A slot can be filled with either data or other objects like *procedures* to perform some action; for example, a procedure to extract or analyze data might be associated with a particular slot.

The state of ES technology today has progressed significantly since its inception nearly 20 years ago. For the CPE domain, many performance rules-of-thumb can be obtained by an extensive literature search. But many of the issues surrounding the development of useful knowledge bases have not yet begun to be addressed by ESes. For example, you have little experience today in determining the certainty values and certainty threshold values that should be associated with rules. Some have proposed obtaining threshold values based on analyzing historical data. Knowledge engineering techniques have been developed that are based on statistical data analysis to help with this task. This leads us to the next section: how to incorporate the analysis of historical data into an ES.

Analysis Of Historical Data

The base set of primitives outlined in the previous section provides a beginning for a set of functions that a comprehensive ES would provide. A key weakness of the existing set of primitives is primarily due to its one-dimensional view of the system; that is, it analyzes the system from a global perspective rather than a more specific *cause/effect* one. In addition, the ES for the most part ignores any existing patterns in the data or historical trends. For example, assume an ES analysis dialog finds that CICS response time is too high. The symptoms that the ES used to reach this diagnosis are:

- CICS is delayed more than 30 percent by CPU queuing
- CICS is a favored workload
- Another workload is dominating the processor.

A real-time monitor would ordinarily be used to discover whether a CICS application is being delayed by higher dispatchable work; and an examination of the Dispatching Priority (DP) of an IPS performance group would exhibit what address space controls the access to the CPU; the more favored the workload, the higher the DP will be. This is a straightforward analysis and though the human performance analyst is responsible for validating the presence/absence of these symptoms, an ES could be engineered to perform the same sequence.

Yet the above analysis is on a rather general level . . . it views the CICS workload as "the center of the universe" in that it does not actually examine *which* other workload is impacting CICS. A human consultant would in all likelihood look closer to see not only *what* workload is causing the degradation to CICS, but might also ask the following questions:

- Have the number of CICS transactions been increasing over time?
- Has the CICS response time degraded over time?
- What workloads have grown recently?

Upon closer examination, the first class of questions that could be asked are *workload* related. Simply stated, *has the workload on the system changed?* And, if so, how *significant* is that change? If the workload *has changed* significantly, there is greater likelihood that the system's tuning parameters are not optimal. The diagnosis made by the ES should supply all that information.

Statistics provide a wealth of mathematics to answer such questions. As an illustrative example, consider a simple workload where the *expected* distribution of work is as follows: CICS accounts for 40 percent of the work on the system, IMS accounts for 30 percent, TSO accounts for 20 percent and batch accounts for the remaining 10 percent. Where does this *expected* distribution come from? In many environments, negotiated service level agreements provide the foundation for such thresholds of the makeup of the distribution. A cluster analysis of historical data collected over several months is often used to provide this type of information. Next, if you collect many samples of the system over a recent period, you might find that CICS accounts for 33 percent of the work on the system, IMS accounts for 37 percent, TSO accounts for 24 percent and batch accounts for six percent. Again, the fundamental question is whether the workload has changed. To answer the question, use the *Chi-Square* goodness of fit test:

$$X^2 = \sum_{j=1}^{k} \frac{(O_j - E_j)^2}{E_j}$$

where there are k workloads, O_j is the *observed* percent for the jth workload and E_j is the *expected* percent for the jth workload. Simply stated, Chi-Square (X^2) assumes that the workload has *not* changed; statistics texts publish a Chi-Square table of *critical values*. You can reject the assumption if the calculated value of X^2 exceeds the published critical value; that is, if the workload *has* changed, X^2 will be larger than the published critical value. Rows in the Chi-Square table are defined by the number of *degrees of freedom (DF)* — for our class of problem, DF = $k-1$ = 4-1 = 3. The Chi-Square table for the first three degrees of freedom is as follows:

DF	$X^2_{.99}$	$X^2_{.95}$	$X^2_{.90}$	$X^2_{.75}$	$X^2_{.50}$
1	6.63	3.84	2.71	1.32	.455
2	9.21	5.99	4.61	2.77	1.39
3	11.3	7.81	6.25	4.11	2.37

Thus, for this problem,

j	Wkld	E_j	O_j
1	CICS	40	33
2	IMS	30	37
3	TSO	20	24
4	Batch	10	6

$$X^2 = \frac{(33-40)^2}{40} + \frac{(37-30)^2}{30} + \frac{(24-20)^2}{20} + \frac{(6-10)^2}{10}$$

Given three degrees of freedom, you can say with better than 75 percent confidence (and probably better than 80 percent) that the workloads are *not* the same; yet, you cannot be *90 percent* certain. Looking closer, you see that the increase in the IMS workload contributed most to the Chi-Square calculation. With this type of *fast analysis* capability, the ES could make a more detailed diagnosis of the high CICS response time. For example: "The characteristics of the workloads currently running on the system have changed significantly. We state this being better than 75 percent confident. IMS was the single largest contributor to this change. Since IMS has a greater dispatching priority than CICS, CICS is unable to obtain the required number of CPE cycles it needs to meet established service level objectives." From this type of diagnosis, the ES can go on to recommend that a capacity planning study should be performed using an analytic modeling package to establish the processor's capacity to run both workloads successfully.

It is exactly this type of *confidence* information that is required for the ES to use fuzzy reasoning. Statistical techniques and corresponding tables have long been in existence and calculations such as that outlined above provide a *mathematical* foundation for calculating confidence values and thresholds.

So how do you extend your existing ES model? You need primitives that are workload related and they must be able to supply answers to the following types of questions:

- Have the workloads changed?
- How significant is the change?
- Do patterns exist in the workload?
- Are the changes in the workload directly related to an increase in business?
- Is there a natural *business cycle* to the workloads?
- Is there missing data and, if so, is it significant?

Techniques such as the Chi-Square, Cluster Analysis, Regression Analysis and Time-Series Analysis can be used to answer these questions but not without some serious thought. For each of these techniques, estimators exist that provide the bounds on the confidence of the answer to the posed question.

A new primitive, generically called *history* is needed that allows the ES to drive the analysis process. *History* must have as one of its parameters a keyword for the type of question to be asked. In addition, you need to supply a description of the measurement variables that should be analyzed, the time interval over which the analysis is to take place and a parameter that will be returned containing the results of the analysis. With this class of primitive, the *rules* making up the knowledge base can take this analysis of historical data into account.

Consider a second possible scenario for the ES to analyze. Again, today's ES technology might diagnose IMS response time as being excessive because it exceeds a given threshold of 80ms. Using historical analysis, the ES could ask the following questions:

- Does IMS transaction response time fluctuate periodically?
- Are the number of CI/CA splits increasing?
- Has the DL/I call pattern changed?
- Has the ratio of physical to logical I/Os increased?
- How frequently is the Database Reorganization utility executed?

All of the above questions are based on examining historical data for patterns. With the proper analysis, a diagnosis of the excessive response time might be:

IMS database IMS001 has periods of low CI/CA splits followed by periods of high counts of splits. The intermediate split process is impacting I/O response time and, therefore, transaction processing time.

The corresponding treatment for the problem might be:

Since the CI/CA split count first increases over time and then decreases, database reorganization is helpful; nevertheless, the database should be reviewed for design deficiencies.

One additional drawback of only using thresholds is that individual variables do not provide information on *interactions* among variables. For example, in VM you could see large page waits with only a moderate page-read rate and a moderate paging path response time. Merely using thresholds is equivalent to asking, *"Is this a good value for this variable?"* This simple question must consider the hardware, software, workload and settings of tuning parameters.

Take a last look at thresholds. R. Orchard from the College of Staten Island developed a fast technique for determining the relationship between computer system variables. If A and B are two system events, you can count the number of times A occurs, the number of times A does *not* occur, the number of times B occurs and the number of times B does *not* occur. For example, let A be "CPU utilization > 90 percent" and B is "Channel Utilization > 30 percent." To find out if these two events are correlated, form an Occurrence Matrix.

Occurrence Matrix

	B	−B
A	a	b
−A	c	d

Here, a is the number of times both event A and event B occurs, b is the number of times event A occurs and event B does not occur and so on. By taking the determinant, $m = ad - bc$, you find that A is related to B if $m > 0$ and A is related to -B if $m < 0$. Thus, for this example, if you sampled the system 50 times (N), where $a=7$, $b=23$, $c=10$ and $d=10$, you find that $m = -160$ implying that the high CPU utilization is related to *low* channel utilization. And this technique can be extended for confidence to be associated with its results: how big does *m* have to be for the relationship to be statistically significant? As before, a Chi-Square calculation will provide a confidence value corresponding to the relationship.

Example:

Let $N_1 = a+c$, $N_2 = b+d$, $N_3 = a+b$, $N_4 = c+d$, and $N = a+b+c+d$, and $m = ab-bc$. Then, Chi-Square is calculated as follows:

$$X^2 = \frac{N * m^2}{N_1 * N_2 * N_3 * N_4}$$

The Chi-Square Table used is the same as that used in the previous Chi-Square example; here, there is only one degree of freedom, since all the samples come from a single population.

Thus,

$$X^2 = \frac{50\ (-160)^2}{17 * 33 * 30 * 20}$$

$$= 3.80 > 2.71 = X^2_{.90}$$

Thus, this Chi-Square calculation indicates with greater than 90 percent confidence that A and B are not independent, but, in fact, are related (high CPU utilization is related to low channel utilization).

For robust knowledge bases to be built that use the *history* primitive, the measurement data must be in a readily accessible form. You need to access not only data of the recent past, but also data that reflects summarized system activity over weeks and perhaps months. Ideally, measurement data needs to be in a single database image that contains data from a wide variety of measurement sources. Commercial products are available with these characteristics (SAS/Merrill Consultants' MXG and LEGENT Corporation's MICS). In an ideal setting, a *relational* database management system could be used to build queries of the performance database dynamically, so "canned" analysis queries do not have to be present *a priori*. Considerable research has been done recently that examines intelligent database interfaces as well as knowledge system architectures. This is clearly the road that comprehensive ES development will embark upon soon.

Summary And Conclusions

ES as applied to problems in the CPE domain is still relatively new. Though the AI/ES technology has finally emerged from academia to industry, building a comprehensive ES is still a complex and difficult task. Though the use of simple measurement thresholds has its place in CPE to trigger tuning studies, the simple ES model of basic primitives needs to be extended to include analyzing historical data as well as recent data. This must be done so basic workload relationships can be verified or refuted. Classic as well as not-so-classic statistics provide us with a rich set of analysis techniques that cannot only determine relationships but can formalize applying confidence values to those relationships. The comprehensive ES should incorporate the use of frames to initiate the analysis techniques and should also use fuzzy reasoning so the confidence values calculated become part of the inferential process. Finally, the use of a performance database as a single logical repository of measurement data from a wide variety of measurement sources over an extended time period will simplify the analyses required. Future work will entail designing intelligent database/ES interfaces as part of a comprehensive knowledge-based system architecture. As these new architectures occur, interim architectures will no doubt be a hybrid ES/human system. For example, ES directing the analyst to performing specific analysis is similar to the role of a CPE consultant working with a staff member on a performance problem. The manager of performance should not look upon this emerging technology as one that will replace human expertise; rather, it is a tool that will make current staff more productive by reducing the time required to analyze problems while providing a formalized technique for examining system problems.

ABOUT THE AUTHORS

Bernard Domanski, Ph.D., has more than 17 years experience in the data processing industry with 13 of these years directly in the computer performance management and capacity planning area. He holds a Ph.D. in computer science from the City University of New York where he is an associate professor in the Computer Science Department at the College of Staten Island. As principal of Domanski Sciences, he developed the first commercially available Expert System.

Sid W. Soberman is senior vice president of Domanski Sciences where he develops knowledge bases for MVS and CICS systems. He has been involved in commercial data processing for IBM environments since 1972. His areas of expertise include performance management and capacity planning for MVS, CICS, IMS and DB2-based systems and he is experienced in the use of CK-ROM technology. Domanski Sciences, 16 Colonial Court, Howell, NY 07731, (201) 367-0257.

VSAM Tuning For Data Entry

By David Shein

A powerful tool offering the application designer many useful options is VSAM, IBM's flagship access method. Like any general purpose software, VSAM was designed to solve a certain class of problems. And like any tool, VSAM performs best on the type of application it was intended for and less well on other types. This article is about the performance problems VSAM exhibits when it is used for data entry (a type of application it was not really designed for) and what can be done to alleviate those problems.

Before continuing, it is essential to define the term *data entry application*. By this I mean an application in which most of the data to be processed is new data being added to the file, not data that is pre-loaded and subsequently just retrieved and/or updated in place. VSAM was principally designed for the latter case and tends to give inferior performance in the former.

A brief description of the problem that triggered the investigation for this article and the application in which the problem surfaced are presented first. Next I will take a minor detour to describe how VSAM handles insertions, since this is where the heart of the problem lies. Finally, I will review a number of file design and tuning options and their effect on performance, making reference to a series of trial runs used to evaluate the various alternatives.

The Problem

The Automobile Club of Southern California has an application that records information gathered by claims adjusters in the field. Frequently, only part of the information needed to process a claim is available at the time of initial data entry. Typically, the user will enter some of the required information and then return at a later time to key in the remainder. The application, therefore, allows entry of partial information that is simply stored in a VSAM dataset until the user signals that data entry is complete. Then and only then is the information released for further processing. For convenience, this dataset will be referred to as the *collector file*.

Each night, the collector file is reinitialized and rebuilt by a batch job. If a transaction is complete (that is, if all required information is present), the transaction is deleted from the collector file and moved elsewhere for further processing. However, if only a partial transaction has been entered, then the information is left in the collector file where it will be available the next day. In this way a user can begin data entry today and return tomorrow to complete the process; the partial information is retained in the collector file until the transaction is complete.

In practice, roughly 95 percent of the information in the collector file is removed each night; only about five percent is held over for the next day. From VSAM's viewpoint, therefore, about 95 percent of the records in the collector file are inserted *after* the initial load. This high (19:1) ratio of new data to old represents a scenario that is relatively atypical for VSAM applications and it produces inferior performance.

VSAM Background

In order to appreciate why large-scale inserts to a VSAM dataset cause difficulties, it is important to understand the structure of a VSAM dataset and, in particular, how inserts are processed. (If you already have a solid understanding of VSAM, feel free to skip this section.)

VSAM supports several different file organizations. By far the most commonly used is the Key Sequenced Dataset or KSDS. In a KSDS, records are stored in ascending sequence based on a *key* field within the record. Since the records are accessed by key, each record's key must be unique. A KSDS is made up of two pieces: an *index* component and a *data* component. The data component holds the actual data records, while the index component contains pointers indicating where the records are located within the file.

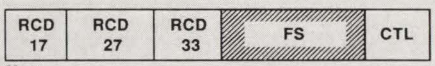
There are several other types of VSAM datasets, but the only one that concerns us is the Entry Sequenced Dataset or ESDS. In an ESDS, records are simply stored in the order received. This is the VSAM equivalent of an ordinary sequential dataset.

For both KSDS and ESDS files, VSAM also allows you to define *alternate indexes*. An alternate index is functionally similar to the index component of a KSDS; it relates key values to record locations. The KSDS or ESDS with which an alternate index is associated is known as the alternate index's *base cluster*. The main value of an alternate index is that it need not use the same key field as does the primary index; thus, an alternate index provides the ability to retrieve records based on more than one key field, depending on the requirements of the application. The most common use of an al-

FIGURE 1

A VSAM dataset consists of a series of CIs.

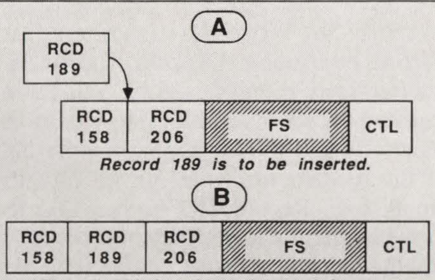
FIGURE 2



Key
CTL - Control Information
FS - Freespace
RCD - Data Record

Internal structure of a VSAM CI.

FIGURE 3



A: Record 189 is to be inserted.

B: Record 189 has been inserted between records 158 and 206.

ternate index is, quite naturally, to provide access to a KSDS by some key other than the prime key. As we shall see, however, alternate indexes have other uses as well.

A VSAM dataset is made up of Control Intervals (CIs). See Figure 1. A CI is roughly analogous to a *block* in older access methods such as BSAM and BDAM. The parallel is not exact, however, for several reasons. First, CI size is almost never an exact multiple of logical record size. Instead, CI size is restricted to just a few possible values. The application designer (we hope!) chooses a CI size that works well for the particular application. A common CI size is 4K (4096 bytes). Second, all CIs within a dataset are the same size. And third, each CI contains, in addition to the data records themselves, control fields which describe the size and location of the records within the CI. Figure 2 shows the internal layout of a CI.

A VSAM file is subdivided into Control Areas (CAs), which like CIs are of uniform size within a file. The maximum (and most common) CA size is one cylinder.

To summarize, data records are contained within CIs; CIs are contained within CAs and one or more CAs constitute a dataset or file.

As you might expect, the records within a KSDS CI are arranged in ascending key sequence. Figure 2 clearly shows this ordering; the number shown within each logical record represents the key value for that record.

When defining a VSAM file, the user can specify the amount of *freespace* to allocate. Freespace is space to be set aside within the file to accommodate future inserts. Obviously, the amount of freespace will have a great bearing on how efficiently records can be added (inserted) after the file is initialized. As with most other characteristics of a VSAM file, freespace is specified in the IDCAMS DEFINE command that creates the file.

Freespace comes in two flavors — CI and CA freespace. They are specified independently of each other.

CI freespace is the percentage of space to be left vacant within each CI when the file is initially loaded. For instance, if the CI size is 4K and 25 percent CI freespace is specified, the initial load will only use 3072 bytes in each CI, leaving 1024 bytes empty.

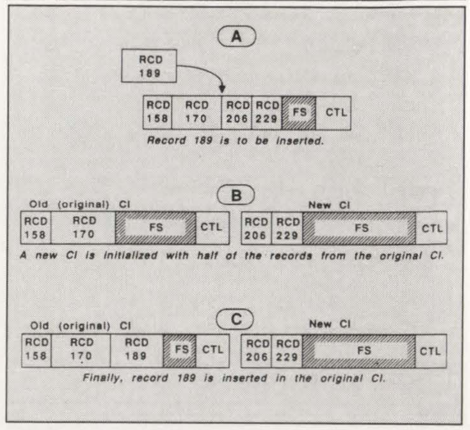
CA freespace is the percentage of the CIs within each CA that are to be left completely empty when the file is first loaded. For example, assume that a VSAM dataset is being defined on 3380 DASD with a CI size of 4K and a CA size of one cylinder. Since a 3380 track will hold 10, 4K blocks and there are 15 tracks to a cylinder, it follows that each CA will hold 150 CIs. If CA freespace is specified as 20 percent, then 30 CIs (20 percent of 150) will be left empty in each CA when the file is loaded.

Now you are ready to look at the way VSAM processes insertions. When adding a record to an existing KSDS, VSAM uses the index to determine (based on the record's key) in which CI it belongs. VSAM then retrieves the CI and checks to see if there is enough unused space (freespace) within the CI to hold the new record. If so, the record is added and the CI is rewritten. This *normal* insert scenario is illustrated in Figure 3. Record 189 is to be inserted in the CI shown. Logically, record 189 fits between records 158 and 206 which are already present. Since the CI contains sufficient space to hold record 189, record 206 is shifted to make room and record 189 is inserted in its proper place.

So far, so good. This process works fine if there is sufficient freespace in the CI. What if there is not?

Figure 4 illustrates what happens when the CI is too full to permit a normal insertion. Here, VSAM needs to insert record 189 but cannot because the CI is full

F I G U R E 4



or nearly so. When this situation arises, VSAM performs a CI *split*. The full CI is effectively divided in two. VSAM locates an empty CI somewhere within the same CA and moves approximately half of the records from the full CI into the empty one. Record 189 is then inserted in the original CI; both CIs are rewritten and the index is updated to reflect the new arrangement. This process works well, but it causes additional I/O since two updated CIs must now be written out to DASD instead of one. CI splits often cause extra index I/O as well.

It is easy to see that excessive CI splitting can seriously degrade performance due to the extra I/O it entails. However, things can get even worse. What if there are not any empty CIs in the CA?

If you have stayed with me this far, you can probably guess what happens — VSAM performs a CA split. This is pretty much what it sounds like: a new CA is initialized and about half of the CIs are moved from the full CA to the new (empty) one. When this process is complete, the original CI split is performed. Because of the extra I/O they generate, CA splits are really bad news. A CA split typically causes dozens of extra reads and writes on top of the (comparatively insignificant) CI split overhead — and all just to insert *one* record! It is an exercise in understatement to say that if performance is important to your application, CA splits should be avoided whenever possible.

The Objective

The problem, in brief, is how to avoid CI and CA splits (or at least hold them to a minimum) for a file that has most of its records added "on the fly" or after the initial load. There are several VSAM options you can use to improve the performance of a data entry file, but they all impose tradeoffs. For example, the more

New for your MVS or VM environment!

Are you under the gun to create documentation?

HERE'S AN EASY WAY TO BITE THE BULLET.

Everyone knows that their Data Center should document important policies, standards, and procedures. The trouble is, with all the crushing pressure of work, documentation always seems to go on the "back burner." These days, that can land you in hot water. With all the strict regulations and tough compliance requirements you face, it becomes *absolutely necessary to take action* and finally solve the "documentation dilemma."

Luckily, we can make compliance easier than you ever dreamed possible!

Introducing the Information Systems Series.™ High caliber documentation the painless way.

The ***Information Systems Series*** documentation framework is designed to help you create comprehensive policies, standards, procedures, and user support materials for your entire MVS or VM Data Center.

Created by highly experienced data processing experts, the ***Information Systems Series*** is an interlinked set of five manuals and accompanying, corresponding diskettes: *The IS Policy Guide, The Data Center User Guide, Systems and Programming Standards, The Operations Guide,* and *The Project Plan and Implementation Guide.* All you have to do is follow the simple, imbedded instructions. In other words, you *personalize* the documentation using your own IBM PC or compatible – which means you can easily integrate any existing documentation quickly and easily!

An important part of the ***Information Systems Series*** is our unique *Project Plan and Implementation Guide* that leads you effortlessly through the entire documentation process. No wonder many forward-looking government, business, banking, medical, and telecommunications IS Departments have put the power of the ***Information Systems Series*** to work.

Demo Diskette

See for yourself what the ***Information Systems Series*** can do for your VM or MVS installation. Call for a Demo Diskette to assist you in your evaluation. Isn't it worth a few moments of your time to prove to yourself that the ***Information Systems Series*** can save you endless hours of work and many thousands of dollars?

Call toll-free 1-800-777-0970

The Computer Resources Group, Inc.
303 Sacramento Street
San Francisco, CA 94111

CIRCLE #69 on Reader Service Card ▲

freespace you allocate (bearing in mind that some portion will never be used), the more split activity you can avoid. In this case, you gain performance at the cost of DASD space.

CI size is another parameter that has a direct effect on performance. The larger the CI size, the fewer splits. (A larger CI requires more records to fill it up.) However, the larger the CI, the more channel and device time needed to read or write it and the more virtual and real storage needed for buffers. In this situation one type of performance is being traded for another. If this is beginning to sound complicated, good. It is.

By now you may be wondering how to determine which combination of parameters is the best for a given application. The most dependable answer is, experiment and see. In the case of the claims application described above, that is exactly what was done; test runs were done with various VSAM file definition options to see which file design gave the best performance. The remainder of this article will focus on the results of those trials.

Design Alternatives And Test Results

The following procedure was used to determine the optimum VSAM definition for our collector file. One day's actual data was acquired via the simple expedient of capturing a copy of the collector file both *before* (at the start of the day) and *after* (at the end). A subset of this data was used for all subsequent test runs.

For each test run, a new VSAM dataset was DEFINEd and loaded with the before data which amounted to 500 records. A batch program was then run to add the remaining data (9500 records) in a random sequence and to retrieve 10 records for every one written. This pattern of access was chosen because it closely mirrors the actual application being simulated.

To assure consistently comparable results (that is, to be certain you compared apples to apples), the same input data in the same sequence was used for all test runs. The results of the trial runs are shown in Table I. The numbers in the table reflect the update runs and do not include the initial DEFINEs and loads.

The design of any VSAM file requires making choices. Selecting VSAM dataset parameters is a process of weighing tradeoffs and choosing to optimize certain values at the expense of others. In the case under discussion, the goal was maximum performance; DASD space utilization is not considered a significant parameter for this dataset. This is because the file is not very large and it is heavily used by on-line (CICS) applications, making performance the principal concern.

Analysis Of Test Results

In the following information, *options* refer to the individual line item results reported in Table I.

Option 1 is the *vanilla* or baseline run that was done to provide a benchmark for evaluating the results of subsequent VSAM parameter changes. Option 1 used a dataset with 4K CIs, no freespace and default buffering (one index buffer and two data buffers).

Option 2 is the same as Option 1 with one difference: two index buffers were allocated at run time (VSAM's default is to use a single index buffer). Having the additional buffer permits VSAM to keep the highest level index record in memory instead of having to retrieve it afresh on each I/O request. As the results demonstrate, this made a dramatic difference. This highlights one of the basic rules of thumb for VSAM datasets: always use at least two index buffers. Doing so costs next to nothing in terms of resources and makes a big difference in performance.

Options 2 through 16 involve various combinations of CI size and freespace values. The trend here is clear: larger CI sizes produce less I/O. As always, however, this must be balanced against the hidden tradeoffs. In this case, one of the tradeoffs is channel connect time. As CI size increases, fewer but larger physical records are transferred between DASD and main storage. Although the figures are not shown in the table, empirical measurement did, in fact, confirm that the channels are more heavily utilized as CI size increases.

Another possible tradeoff involves DASD space. Some of the CI sizes tested result in poor DASD track utilization. As mentioned earlier, space was not a consideration for this particular dataset. If the file were large, however, it might become a factor.

Options 17 and 18 represent an attempt to eliminate split activity completely by using a non-KSDS dataset organization. An ESDS was defined and an alternate index was associated with it to permit retrieval of records. (The alternate index used the same key field as the original KSDS.) The idea was that CI splits would be avoided thanks to the strictly sequential nature of I/O tc an ESDS. As the table shows, the results were disappointing. The additional overhead of the alternate index far outweighed the savings realized by the non-KSDS dataset organization.

Option 19 represents a non-VSAM solution and requires some explanation. The Innovation Access Method (IAM) is a product of Innovation Data Processing (Little Falls, NJ). IAM is an access method that provides the same functions as a VSAM KSDS but with significantly greater efficiency. Without indulging in a lot of fine tuning, one test case was run using IAM instead of VSAM. The results speak for themselves: IAM's performance was superior to any of the VSAM alternatives. Not surprisingly, IAM was the access method that was ultimately used for the collector file.

Something further needs to be said about these results. The update program used for this investigation, no matter how carefully designed, is only a simulation of the actual application. The real application runs in an on-line transaction proc-

TABLE 1

VSAM Data Entry Tuning Test Results

Option	File Type	CI Size	Freespace (CI/CA)	CPU Seconds	EXCPs
1	KSDS*	4K	0/0	114.56	284,081
2	KSDS	4K	0/0	65.61	73,513
3	KSDS	4K	50/50	65.88	74,106
4	KSDS	4K	99/90	65.49	72,242
5	KSDS	8K	0/0	61.24	59,683
6	KSDS	8K	50/50	61.66	59,131
7	KSDS	8K	99/90	62.29	59,467
8	KSDS	10K	0/0	61.47	56,368
9	KSDS	10K	50/50	61.54	56,332
10	KSDS	10K	99/90	60.80	57,408
11	KSDS	12K	0/0	60.04	54,360
12	KSDS	12K	50/50	59.76	54,402
13	KSDS	12K	99/90	60.01	55,774
14	KSDS	22K	0/0	61.08	50,328
15	KSDS	22K	50/50	63.54	50,227
16	KSDS	22K	99/90	63.90	53,853
17	ESDS**	8K	N/A	108.02	133,367
18	ESDS**	22K	N/A	113.24	133,454
19	IAM	N/A	N/A	50.16	33,168

Notes:

*Default buffers (1 index, 2 data)
**With an Alternate Index.

essing environment and is just one of a great many applications sharing that system and competing for resources. On-line workloads fluctuate in real time in an unpredictable manner. Sooner or later, the file tuning choices you make must be subjected to the acid test: trying them out in a live system. Ultimately, that is the only way to know for certain how they will perform.

Conclusions

Because of the way VSAM handles additions to an existing dataset, VSAM applications that do large numbers of random insertions pay a significant penalty in terms of performance. This impact can be reduced through careful design and tuning of the VSAM dataset.

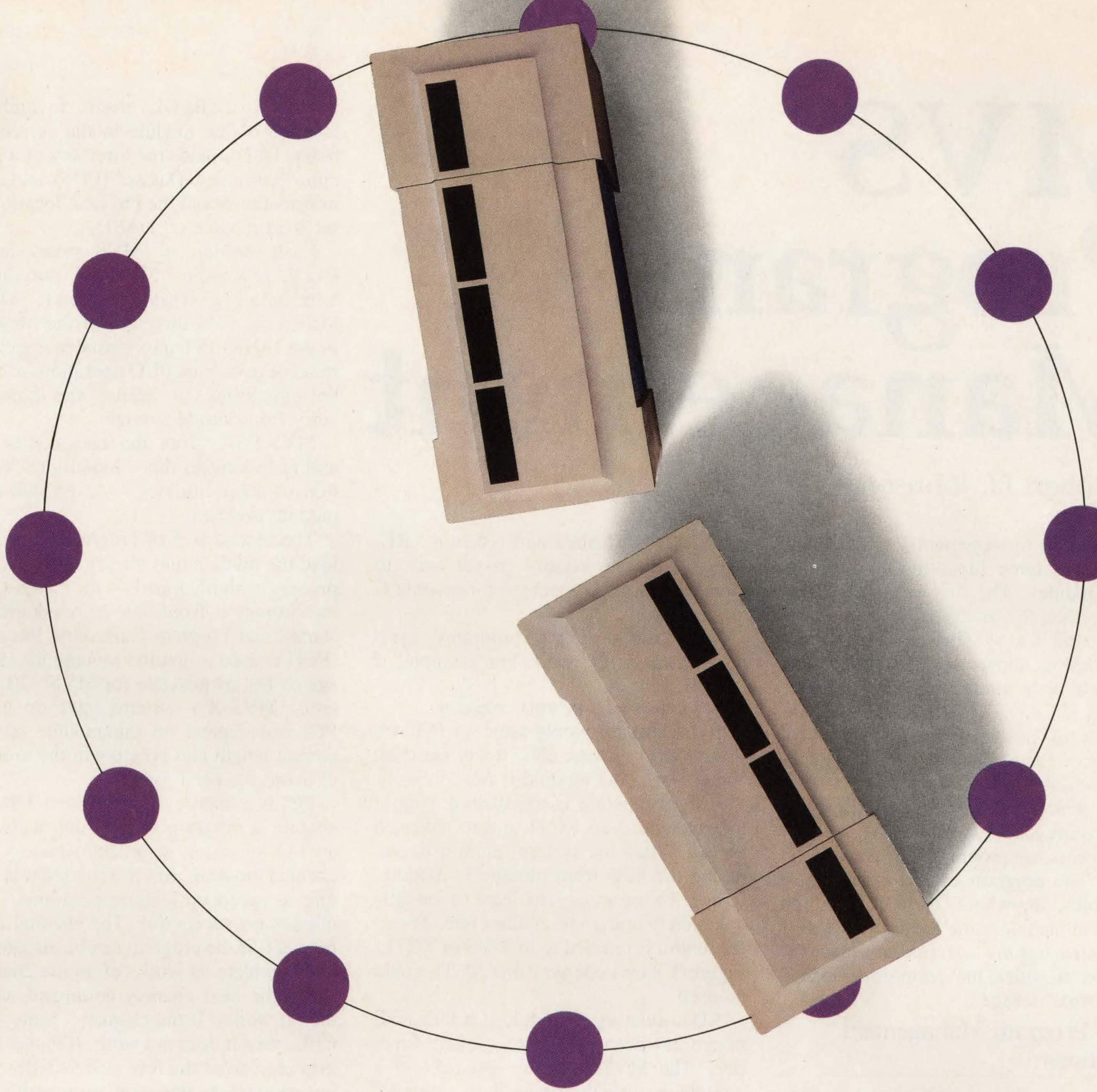
VSAM dataset tuning is an iterative process that requires making choices. Resource tradeoffs must be resolved in the most appropriate fashion considering the nature and sensitivity of the application as well as the resource constraints imposed by the environment. Numerous combinations may have to be tried in order to identify the optimum dataset parameters where *optimum* is defined by the tradeoff decisions that were made. Test runs should be designed to mirror the actual application and environment as closely as possible. In some cases, particularly on-line applications with their unpredictable workload mixes, trying out the most likely combinations in a production environment may be the only sure way to determine what works best.

Products such as IAM offer the designer additional alternatives and can help a great deal. If you have such software available, by all means take advantage of it. However, not every shop has such tools, and even those that do generally have some applications which are not suitable candidates for their use. For example, as this is written, the current release of IAM does not support alternate indexes. If the application requires one, IAM cannot help. Thus, VSAM dataset tuning remains a vital concern.

Optimizing a VSAM dataset requires time, effort and an understanding of how VSAM datasets are structured and accessed. Above all, it requires that design decisions be based on empirical data to the maximum possible extent; there is no substitute for real world testing.

ABOUT THE AUTHOR

David Shein is a systems programmer with the Automobile Club of Southern California (Costa Mesa). The technical staff to which he belongs provides system software support and maintenance, capacity planning, performance management and DASD management. The environment supported includes multiple IBM mainframes, MVS/XA, CICS and a 3000-terminal network.



MVS Program Management

By Robert H. Johnson

Program management for MVS performs three functions for program modules. The first is to search for and schedule the module to be in storage. The second is to synchronize exits to routines during supervisor functions. The third is to fetch modules into storage after a LINK, LOAD, XCTL or ATTACH supervisor call has been issued. This article is a discussion of the searching and fetching functions.

The third function is accomplished by a *relocating loader,* which is the basis for the success of program management in MVS. The program can be compiled or assembled anywhere (usually location zero) and the relocating loader will *change the contents of the program* to reflect the location at which the program is loaded into virtual storage.

MVS Program Management Functions

MVS program management functions consist of several general functions.

One function is finding and *loading* a module into virtual storage if a LOAD macro has been issued. Figure 1 shows a load process. Module A issues a LOAD macro and the address of the module is returned in a general purpose register. The LOAD macro generates a Supervisor Call (SVC) 8. The program must save the address of the module to be used when the program needs to go to the module with a CALL macro. *Module B does not get control until the CALL macro branches to it.*

Another function is linking to a module if a LINK macro has been issued. Figure 2 shows a LINK operation. Module A issues a "LINK" SVC 6 causing module B to be found or loaded and execution begins immediately in module B. When module B is finished and issues a "RETURN" macro, control passes back to module A. This is repeated for module C in the figure.

JES initiators link to programs specified on the EXEC card. For example, if the JCL said

```
//stepname EXEC PGM=myprogrm
```

then the initiator would issue a LINK for "myprogrm." *Note:* SVC 6 will cause an abend 806 if the module is not found.

A third function is transferring control to a module if an XCTL macro has been issued. When the systems analyst is preparing the flow from module to module, it may be necessary to leave a module completely and go to another one. Transfer control is referred to in MVS as XCTL. Figure 3 shows one way that XCTL could be used.

Also, deleting a module if a DELETE macro has been issued is a general function. The MVS way to "get rid" of a module in virtual storage is to "delete" the module. In Figure 1, the program issued a LOAD macro which loaded the program into virtual storage. After all the possible "calls" have been performed, a DELETE macro should be issued to clean up virtual storage.

Allowing modules to establish entry points that can be used if a LINK, LOAD or XCTL is requested is the last function. The program issues an IDENTIFY macro and points to the area that will receive control from program fetch. The program can have the linkage editor create additional "names" for a program. The linkage editor "ALIAS" command is used.

Program Fetch

Program management invokes *program fetch* to bring modules into storage. The first step in program fetch is to issue a Build List (BLDL) macro to find the location of the module in the correct library. BLDL reads the directory of a specific Partitioned Dataset (PDS) and gets information about the physical location of the load module on DASD.

Each version of MVS gives better BLDL processing. MVS/XA introduced LINKLIST Lookaside (LLA), which loaded the PDS directory of the libraries in the LINKLIST into virtual storage. Instead of performing I/O operations to read the directories, it "reads" the directory entry from virtual storage.

MVS/ESA gives the data center the ability to expand this "lookaside" function to other libraries — even non-load module libraries.

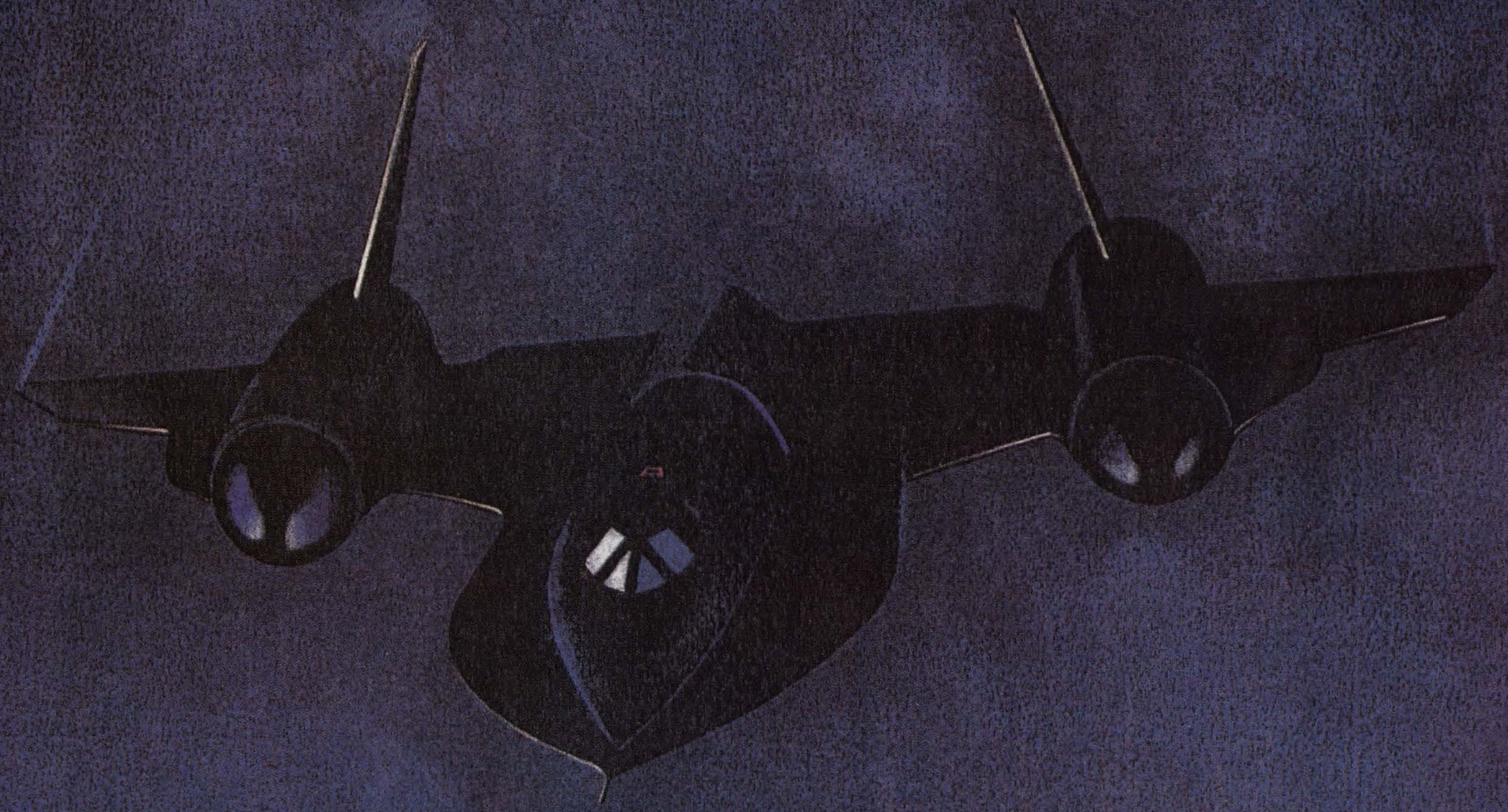
The second part of program fetch is to load the module into virtual storage. This process is highly tuned — up to 64K Central Storage is fixed, one I/O operation is started and Program Controlled Interrupt (PCI) is used to get the module into storage as fast as possible for MVS/370 systems. MVS/XA systems gave up using PCI and depend on the module having correct length and pointers in the module to avoid the need for PCI.

PCI is a feature of the System/370 and enables a program to get control *during* an I/O operation in order to modify the channel program. MVS/370 used this feature to speed up loading programs. It is not always successful. The channel does not wait. If the program can be dispatched and complete its work before the channel needs the next channel command word, then it works. If the channel "beats" the CPU, then it does not work. Relative timings are one of the few aspects IBM *does not* promise as upwardly compatible, so depending on it begs trouble.

Relocating Loader

Figure 4 shows the processing done for loading a program. A particular application or program is developed in *source code* then *assembled* or *compiled* into language which the System/370 understands. The output of the compiler is called the *object code,* which is the instructions in the source converted to System/370 machine instructions in binary codes. The linkage editor converts object code to executable load modules. The program can then be loaded into the computer to execute the instructions. There are three types of loading which are possible.

The first type is *absolute* loading in which the program, at source and compile time, identifies exactly where in the com-



PHAST

Phase Linear's PLSORT is unquestionably the IBM world's fastest sort/merge. Fast and furious on your mainframe system. It streaks through the eye of your production storm hours before the competition. And its speed will allow you to take on a processing schedule larger than any your system has ever weathered.

Not only is PLSORT fast, but you'll be amazed at how fast you can get it up and running on your system. In less than an hour, you'll learn the real meaning of the word, Phast.

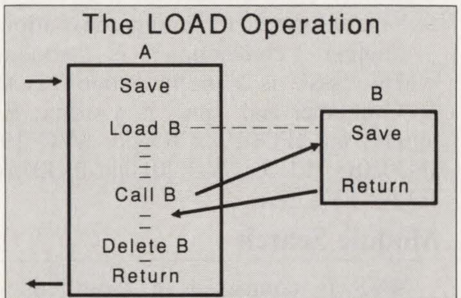
Call us today for a free trial. **1-800-862-SORT • Canada: 1-800-635-0571**

CIRCLE #78 on Reader Service Card ▲

PHASE LINEAR SYSTEMS INCORPORATED
AN ACR/TRITON COMPANY
9300 Lee Highway, Fairfax, Virginia 22031-1207

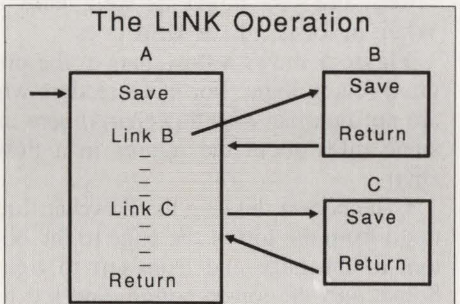
SORT OS • PLSORT VM • PLSORT OS/XA-ESA • PLSORT VMX • EPS • PLSORT OS • PLSORT VM • PLSORT OS/XA-ESA • PLSORT VMX • EPS • PLSORT

FIGURE 1



Module A can issue the MVS macro LOAD, which causes program B to be brought into virtual storage. The address of module B is returned to program A in a general purpose register (zero). The DELETE macro frees the storage occupied by the module.

FIGURE 2



Module A links to module B. After returning from module B, module A does some more processing and links to module C.

puter the program will execute. For example, the source code could specify a particular address, such as x'4000', as the start of the program. The drawback of this technique is that the program must be loaded at x'4000' in the computer to be able to be executed. If two programs have the same address, then they cannot be run at the same time. Early VSE operating systems used this technique because there was a limit of two programs running at any one time: foreground and background. MVS does not use absolute loading.

Another type is *static relocation* loading in which the loader does some work as it is bringing in the program to *change the contents of it* by relocating it to the location in the computer where the program was being loaded. For example, the source code could have an address such as x'4000' as the start of the program, but the loader could change the start address to x'9000' by adding x'5000' to all the places in the program that addressed data. This method's drawback is that once the program is loaded and relocated, it could not easily be moved.

Dynamic relocation, an expansion of static relocation, is the third type. The translation of program references to Central Storage addresses is *delayed until the last possible moment*. A relocating loader changes the contents of a program based on the virtual address that the program is loaded into and the instructions are System/370 instruction formats using base registers to point to 4096-byte pages and displacements within the page.

In Figure 4, the source for a program contains the address of a counter (called ADDCOUNT) within the program. The counter is 500 bytes into the program. After compiling, linking and loading, the area called ADDCOUNT has been changed from 500 to 10500 because the program was loaded at location 10000. The loader went into the module, found the pointer to the counter and added the address that the program was loaded to the pointer. (This example is running under MVS/XA, so the program might be loaded at location 10000. In MVS/XA, the nucleus straddles the 16MB line.)

Remember, a program only references virtual storage addresses. It can be interrupted at any point, swapped out by the Auxiliary Storage Manager (ASM) and reloaded (swapped in) into a different set of central storage addresses without modification.

Memory allocation is performed by the loader while loading in a program or combination of programs. The particular combination of programs can be decided *just prior* to loading.

Where Are The Modules?

When a batch job or TSO session is executing, many MVS modules are needed to perform the services (OPEN, CLOSE, READ, WRITE and others). MVS modules reside in only a few load libraries. (User or application modules may, of course, reside in any load library.)

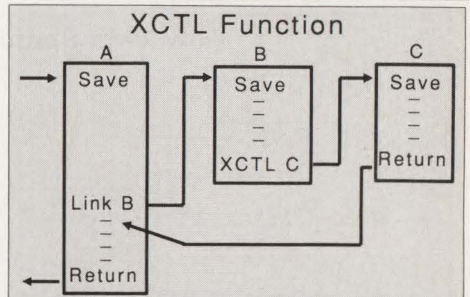
SYS1.LPALIB

SYS1.LPALIB contains the most frequently accessed of the modules which the system and the end user need to execute system functions. I/O drivers, SVC modules and TSO commands are here. Every module in LPALIB is available to a program to load and call.

SYS1.LINKLIB

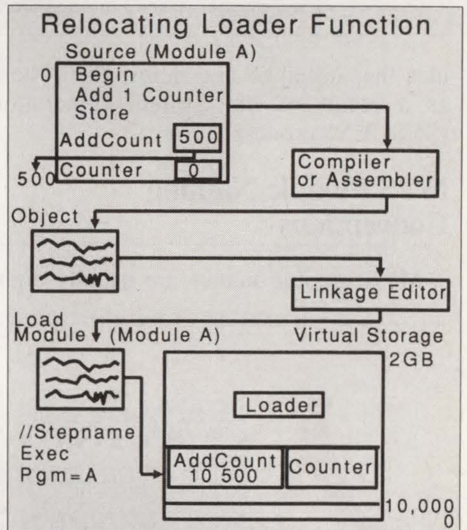
SYS1.LINKLIB contains most of the utilities and the modules which are either not re-entrant or are not normally shared by more than one user at a time. Examples are SORT modules, COBOL and PL/I compilers and IEBGENER. SYS1.LINKLIB is the first of several load libraries which are collectively called

FIGURE 3



In MVS, one module can transfer control (XCTL) to another module. In this example, module A has linked to module B, but in module B it is determined module C must finish the process. AN XCTL macro is issued with loads and transfers control to module C. Return is back to module A.

FIGURE 4



The source for program A is compiled and transformed into object code. The MVS Linkage Editor reads the object code and creates a load module. The initiator loads program A, which causes the contents of program A to be altered based on the point in virtual storage where program A is loaded.

LINKLIST libraries. The data center must specify SYS1.LINKLIB, a library for modules written or maintained by the data center and called SYS2.LINKLIB, for example, and other libraries. The compilers and sort libraries are examples of these "other" libraries.

SYS1.SVCLIB

SYS1.SVCLIB was an important library in MVT systems, but today only contains exit or "appendage" routines. It is kept around mostly because of compatibility reasons.

SYS1.CMDLIB

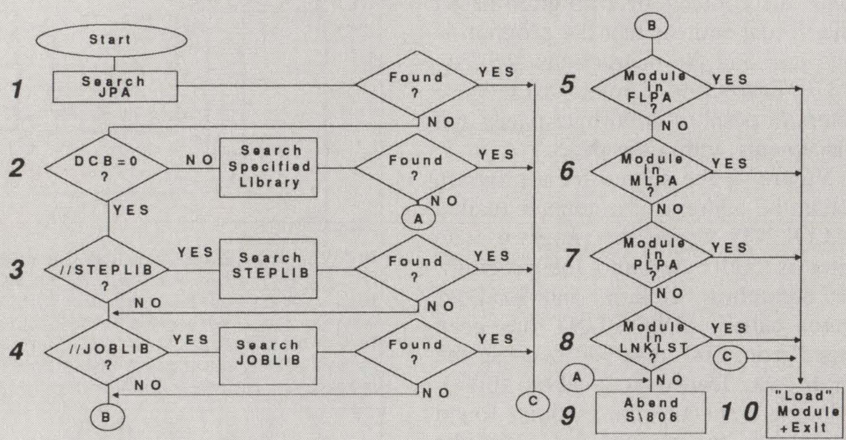
SYS1.CMDLIB contains the TSO command processor modules.

SYS1.NUCLEUS

SYS1.NUCLEUS contains the mod-

FIGURE 5

How MVS Searches For A Module



The flowchart traces the decisions MVS uses to find and/or load a module.

ules that initialize and define the system as a result of the System Generation (SYSGEN) process.

MVS Module Naming Conventions

MVS module names are usually eight characters long. The eight characters may be of two formats. One is sssnnvvv, where "sss" is the subsystem prefix with which the module is associated. For example, if "sss" is "IEF," then the modules are job allocation modules. The middle "nn" is usually a version number or submodule name and the "vvv" is the SVC number or other designation. It is easy to see that, with only eight characters, it is difficult to have informative naming conventions.

Another "convention" is sssooonn, where "sss" is a prefix, "ooo" is the SVC number and "nn" is a suffix. Examples are IFG019xx (Open: SVC 19), IFG020xx (Close: SVC 20) and IFG055xx (EOV: SVC 55).

Module Search

MVS is composed of thousands of modules. There are tens of thousands of messages. There are tens of thousands of abend codes. *There is a madness in this method.* This is a naming convention which dates back to 1965. It is not a good naming convention, is not always followed, but you might as well learn it. When in Rome, do as IBM does.

Figure 5 shows a flowchart of the program search logic. For those readers who are not familiar with flowcharts, here are some rules about the figures in a flowchart.

• The person drawing the flowchart tries to go from the top of the page to the bottom of the page and from left to right. Sometimes the top-to-bottom and left-to-

Give Your Job Scheduler A Good Talking To!

And your Performance Monitor, And your Problem Tracking System, And your Change Control Tool, And your...

Datacenter operations just got a lot easier. With the newest feature of OPS/MVS®, the External Product Interface (EPI™), you no longer have to run your VTAM, 3270-based datacenter management tools manually. Now you can let the EPI tell all your VTAM-based tools what you need them to do. Then, you let the EPI tell you what happened by telling it to send you (and any or all of your support personnel) an EMAIL message. Any application that can be monitored and controlled by a human operator sitting at a VTAM 3270 display (such as CA-7®, OMEGAMON® or NetView™) can be operated by a REXX program via EPI.

The EPI is a standard feature of the OPS/MVS base product (no separate install required) and is *available at no additional cost.* For more information call MVS Software at **213-578-1147**. Our fax number is 213-578-5614.

OPS/MVS: The Most Powerful Automated Operations Tool Is Also the Easiest to Use.

MVS SOFTWARE, INC.

The Automated System Operations Specialists™

"NetView" is a trademark of the International Business Machines Corporation. "CA-7" is a registered trademark of Computer Associates. "OMEGAMON" is a registered trademark of Candle Corporation.

CIRCLE #31 on Reader Service Card ▲

right is reversed if it makes the figure easier to read.

• Ovals (START) are entry points to the figure.

• Rectangular boxes are processing boxes. The contents of the rectangle describe the type of processing. In the first example, "Search JPA" indicates MVS will search the control blocks that make up the JPA. If the module name is found, the address will be loaded and the flowchart says to proceed to circle with a "C" inside.

• Circles with arrows pointing into them are entry points from other areas on the flowchart. Circles with arrows pointing out of them are "branches" or "jumps" to other areas on the flowchart.

• A diamond with arrows coming into and from the points is a decision block. The content of the diamond is a question that can be answered, usually with a "yes" or "no."

Figure 5 shows the order in which MVS program management searches for a load module:

• Job Pack Area (JPA) — If the module has already been loaded and can be reused, return the address of that module.

• DCB$\neq$0 — A load command may specify a specific library that must be used to find the module. If the library contains the module, the module is "loaded" into storage and the address is returned. If the module is not found, terminate the request — the user asked to use a specific library.

• Step library — If there is a STEPLIB DD card search in that library (or concatenated libraries) for the module and if it is found, load it and quit searching. If it is not found, go on to the next step.

• Job library — If there is a JOBLIB DD card, MVS searches the libraries that are pointed to by the JCL.

• Fixed Link Pack Area (FLPA) — MVS searches modules that are loaded into virtual storage and then have their pages locked into central storage. The control blocks searched are called the LPA Queue. If the module is found, its address is returned.

• Modified Link Pack Area (MLPA) — MVS searches modules that are loaded into virtual storage by the SYS1.PARMLIB parameters. MLPA modules are also on the LPA Queue.

• Pageable Link Pack Area (PLPA) — MVS searches modules that are loaded into virtual storage by MVS that are in the SYS1.LPALIB dataset (or other partitioned datasets for MVS/XA 2.2 and MVS/ESA). The searching is accomplished by inspecting the Link Pack Directory containing Link Pack Directory Entries (LPDEs).

• LNKLIST — Member LNKLSTxx of SYS1.PARMLIB contains the names of the datasets that the data center wants to use as a "last resort" to find programs that are requested.

• If the module is not found, the program is ABENDed with a code of 806.

• If the module is found, the module is "loaded" and control is returned to the calling program.

Now turn to some of the above areas to understand when MVS would find a module in these areas.

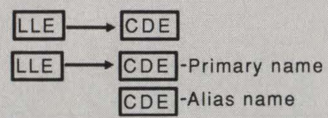
Job Pack Area

The following elements comprise the JPA. See Figure 6.

• Modules already loaded — Virtual storage contains the modules loaded by this address space. The JPA modules reside in the private area of an address space and these programs are only available to this ASID.

F I G U R E 6

Control Blocks For Modules In The JPA



The control blocks used by MVS are the Load LIst Element (LLE), which contains pointers to the Contents Directory Entry (CDE), which contains the module name. In this example, one module has a second name or alias.

• The JPA Queue is a chain of control blocks called Contents Directory Entries (CDEs). Each CDE contains the name of the module, the address of the entry point for the name and use count (how many current users are "in" the module). If the module has an alias (another entry point), then there are two CDEs — one for the "major" name (first name of the module) and one for the "minor" name (alias or secondary entry point to the module).

• The load list is a chain of control blocks called Load List Elements (LLEs). There is an LLE and a CDE for each module that has been explicitly loaded. This queue of modules available within the address space is the first queue the load routine searches.

• The LPDEs represent a particular load module that is loaded into the PLPA.

Link Pack Area

Five elements comprise the LPAs. See Figure 7.

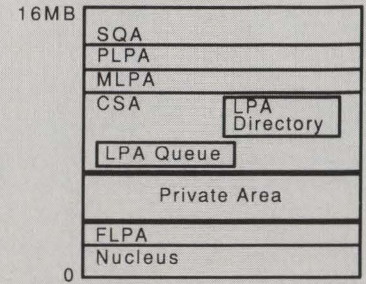
The first element is the Pageable LPA referring to virtual storage containing all the modules which reside in the 'SYS1.LPALIB' dataset.

As the name indicates, the virtual storage occupied by PLPA is pageable — the module may be in central storage or it may be on auxiliary storage. *They are not paged out*. MVS loads the PLPA page dataset at IPL time (when CLPA — Create LPA — is specified) and the pages are only paged in if needed. Referenced pages remain in central storage because the central storage page is "referenced" often.

Large virtual storage constraint relief can be realized by cleaning out 'SYS1.LPALIB'. If the systems programmer has old versions of modules which have been changed (usually names like "OLD0001C" or "ZZZ0001C"), then virtual storage is being wasted by having two copies of these modules in it.

F I G U R E 7

Link Pack Area



The LPA is really a combination of areas built by MVS as it is initializing and control blocks which hold pointers to the modules.

Whole unused functions can still be in the LPA. If the data center does not have any use for ISAM, the ISAM modules can be removed. For example, module IGC019GA is a QISAM load function module. It is about 4000 bytes. If this module were removed from LPA, then your system would not have to load and pass over it for all module search activity. Complete discussion of this and other modules you may want to remove is in *MVS Virtual Storage Tuning Cookbook* (G320-0597).

The second element is the LPA Directory, a chain of control blocks called LPDEs. Each LPDE contains an entry point for each LPA module and alias name.

The Modified LPA, the third element, is an optional list of modules which are included in the LPA for performance or other reasons. MLPA was designed to allow the data center to "temporarily" add modules for the duration of an IPL without having to reload the PLPA.

At IPL time, "MLPA=nn" is specified either by the operator or in member IEASYSxx of SYS1.PARMLIB to tell MVS which modules to load. Use MLPA with caution. If the module belongs in LPA, put it there. Some uses that data centers have made for MLPA specifications are:

- Reduction of paging. If several DB/DC systems (such as CICS or IMS) are running, then making one copy of re-entrant routines available to all of the DB/DC jobs saves virtual storage and paging resources. For example, all CICS modules are usually in one program library. If the active CICS modules are moved to the LPA (or

DOCS™

THE BEST-KEPT SECRET IN VSE CONSOLE AUTOMATION.

DOCS is the best-kept secret in VSE console automation because once it's installed, DOCS becomes the backbone of your overall automation scheme. We've got more than 500 satisfied SMARTECH customers that agree.

TOTAL MESSAGE CONTROL

DOCS gives you total control of all console messages because it actually operates as the VSE console. What's more, DOCS provides all the features and information you'll need to automate and manage your local environment.

- Auto-reply
- Message suppression
- Message routing
- Programmable function keys
- Multiple VSE consoles on one CRT
- Remote consoles through VTAM
- Multiple consoles
- Last line/screen recall
- No online dependency
- Keystroke automation

NO RISK, MONEY-BACK GUARANTEE

Buying DOCS is risk free because it comes with an unconditional six-month money-back guarantee. And all it takes is a simple 30-minute installation.

So if you are looking for the secret to VSE console automation, find out what hundreds of our customers already know. For more information, call **1-800-53-SMART**. (Outside the U.S., call 214/956-8324.)

SMARTECH SYSTEMS, INC.

Turning high technology into *SMART TECH*nology.℠

10015 W. Technology Blvd., Dallas, Texas 75220
FAX: 214/357-6338 Telex: 9102503110

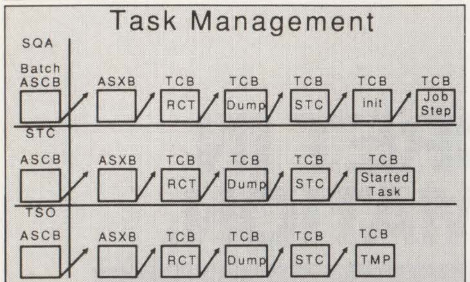
For information about our international representative network, contact SMARTECH SYSTEMS, Inc.

ICP MILLION DOLLAR AWARDS $10,000,000 WINNER

IBM Business Partner

CIRCLE #65 on Reader Service Card ▲

F I G U R E 8



The three types of work - batch jobs, Started Tasks and TSO sessions - all have TCBs associated with each piece required to give control to the user program.

selected from alternate LPA libraries with certain versions of MVS) and CICS is instructed to use the LPA members, then all CICS modules will use the same copy of the module in common storage.

- Add modules for performance reasons. A common practice is to add TSO command modules so that TSO performance is improved. This reduces search time for often-used modules.
- 308x Processor Complex and XA migration. The pageable LPA is protected from modification by the hardware. Many IBM, accounting and other data center modules have violated re-entrant requirements. The MLPA is outside the LPA protection. Use of the MLPA should be a temporary work-around for system abend 0C4s. The modules should be permanently fixed and moved to PLPA.

The fixed LPA, the fourth, is an optional LPA extension. These modules do not reside in the upper part of virtual storage (MVS 370) but down with the nucleus. Many DB/DC SVC modules require fixed LPA residence because they are old code written for OS/MVT. In OS/MVT, all storage was fixed and real.

The fifth element is the LPA Queue or active LPA Queue that is a queue of CDEs modules in the FLPA, MLPA and currently active PLPA modules. It is designed to reduce the search time to find a module.

Auxiliary Storage Libraries

Auxiliary storage libraries are load libraries containing modules that are not in virtual storage. These are the Linklist libraries and job and step libraries.

Program Properties Table (PPT)

The Program Properties Table (PPT) is where the installation can assign special properties to programs by placing their names in this table. The program can be given special attributes such as noncancelable, different storage keys and be marked not swappable.

Starting with Release 2.0 of MVS/XA, the data center can use a member of SYS1.PARMLIB (SCHEDxx) to modify the PPT to suit its requirements.

Task Management

MVS has two types of tasks: the Task Control Blocks (TCBs) and the Service Request Blocks (SRBs). TCBs represent work within an address space such as user programs, utilities and system programs which are operating on behalf of, or performed for, the user task. SRBs represent work requested by one address space that is executed in another address space.

TCBs are created when the user explicitly or implicitly issues an ATTACH macro. Look at Figure 8. Three types of work are shown — batch jobs, Started Tasks and TSO address spaces. The Address Space Control Block (ASCB) points to an Address Space Control Block Extension (ASXB) in the Private portion of virtual storage. The ASCBs are in the System Queue Area (SQA) because all ASCBs must be available to all address spaces.

In the example, there are five programs "running." The Region Control Task controls the address space. The DUMP task takes over if any abend is scheduled. The Started Task Control (STC) module is executed when the operator issues a start for the initiator. The JES Initiator code is linked to and, finally, the program called for in the job step is given control.

For a started task other than the initiator everything is the same, except control is transferred to the program specified on the EXEC card of the started task procedure.

In a TSO session, the Terminal Monitor Program (TMP) is given control to execute the user's TSO commands.

How did all these tasks get started? What MVS function is controlling them? Task management performs services for both problem (that is, application) and system programs. The MVS Assembler macro names are included in the following list of MVS task management functional areas:

1. Create and delete subtasks: (ATTACH/DETACH).
2. Control the execution of a task.
   a. Changing the dispatching priority of a subtask (CHAP).
   b. Allowing a task to wait for an event (WAIT, EVENTS).
   c. Notify another task of a completion of an event (POST).
   d. Serialization of tasks (ENQ, DEQ, RESERVE).
   e. Provide program interruption interception (SPIE, STAE, ESTAE).
3. Provide informational services (EXTRACT, TESTAUTH).

Summary

MVS program management locates and schedules modules for execution, synchronizes exits during supervisor functions and fetches other modules into storage on request. The LOAD, CALL, LINK, XCTL, DELETE and IDENTIFY Assembler macros invoke these functions.

Program relocation may be accomplished at several points: by a linkage editor after source program compilation or assembly, by the loader during program fetch or by the program itself during its execution.

Program fetch obtains load modules from several sources: the LPALIB, LINKLIB, SVCLIB, CMDLIB, NUCLEUS library and application libraries, all of which are PDSes. MVS has an algorithm for locating a given module name in the libraries and in shared areas of virtual storage (JPA and LPA, for example). The PPT can give programs operational properties.

Task management manipulates TCBs, representing work in an address space, and SRBs, representing requests for work from other address spaces. The ATTACH, DETACH, CHAP, WAIT, EVENTS, POST, ENQ, DEQ, RESERVE, SPIE, STAE, ESTAE, EXTRACT and TESTAUTH Assembler macros obtain task management services.

ABOUT THE AUTHOR

Robert H. Johnson has been working with IBM operating systems since 1968. He has worked for several large government and private companies either writing large full text information retrieval systems or writing software monitors. Johnson has been a widely acclaimed speaker at SHARE, GUIDE, CMG and at seminars worldwide. This article is adapted from his book titled MVS: CONCEPTS AND FACILITIES *(ISBN 0-07-032673-8) published by the McGraw-Hill Publishing Company, 1221 Avenue of the Americas, New York, NY 10020, (800) 2-MCGRAW.*

Has your staff been very good this year?

Happy Holidays from the folks at Empact Software

EMPACT software
A Boole & Babbage Company
1803 Overlake Drive, Conyers, Georgia 30208
Telephone (404) 483-8852 / Fax (404) 388-9453

CIRCLE #119 on Reader Service Card ▲

Overcoming CPU Constraint In Large CICS Systems

By Ted C. Keller

During the past several years, the CICS product has been continually enhanced. Most of the more significant performance constraints have been overcome in recent releases. At the present time, only a few major limitations remain, the most prominent being CPU constraint. CICS' current architecture limits the amount of application processing that can be performed in a single CICS address space. For many installations, this limitation is one of the principle causes of CICS response degradation. In this article, I will examine how CPU constraint can affect CICS performance and how you can relieve its causes.

The Nature Of CPU Constraint In CICS

CICS, as it is structured today, normally will not provide acceptable service at higher levels of CPU utilization. Unlike TSO, IMS or batch applications which run under separate MVS Task Control Blocks (TCBs) and which are interrupt driven, most CICS work runs under a single MVS TCB and is *not* interrupt driven. The difference in these structures allows TSO systems to function acceptably even when processor utilization is 95 percent or more. In contrast, CICS systems will begin to experience processor-related constraint when a region attempts to use as little as 50 percent of a processor.

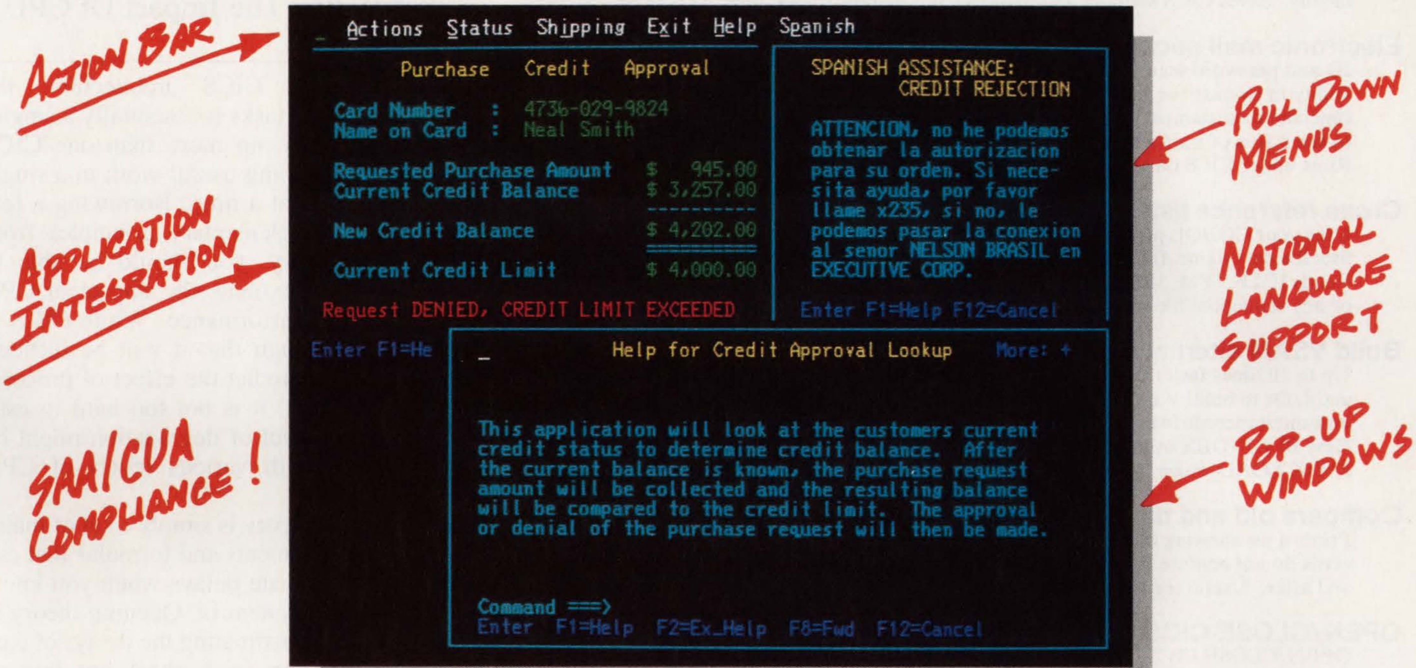
The key to understanding CICS CPU constraint is *CPU demand*. CPU demand is the percent of time that a CICS region is either using or attempting to use a processor. It includes not only the time CICS is actively executing instructions, but also the time it is prevented from doing so by other factors. For example, when CICS experiences a page fault, the CPU cannot be utilized until the page fault is satisfied. Assuming a page service time of 25ms to external storage, a paging rate of two page-ins per second would make it impossible to utilize the CPU 50ms per second or five percent of the time. Similarly, factors such as VSAM Control Interval (CI) and Control Area (CA) splits (prior to CICS Release 2.1.1) can interrupt processing and restrict access to the CPU while the splits are being serviced.

CPU demand also includes the impact of higher priority MVS tasks. CICS is not normally the highest priority task in the operating system and, even in a multiprocessor environment, will experience some delay waiting for higher priority tasks. This time will increase as total CPU utilization increases. In essence, CPU demand is all the time the CICS TCB is not voluntarily waiting but is trying in any way to make use of a processor. CPU demand is the best indicator of CICS CPU constraint.

IBM has lent legitimacy to the use of CPU demand for determining CPU constraint. In CICS Release 2.1.1, the VSAM subtask monitors CPU demand in the main CICS TCB. When CPU demand is low, random VSAM read requests will be serviced directly under the main TCB. When CPU demand is higher (above 70 percent), VSAM processing for these requests will be passed to the VSAM subtask. CPU demand is being used to determine the point of CPU constraint at which some processing is switched to the VSAM subtask.

A second factor that has a major impact on CICS CPU processing is the length of time tasks retain control of the CPU before returning control to CICS. Since CICS is not interrupt driven, once CICS gives control to a task it cannot accomplish any other work until that task returns control. Tasks which perform extensive calculations or processing can retain control for prolonged periods of time and prevent CICS from dispatching other useful work. No other tasks will be dispatched, I/O events which complete will not be recognized and requests from other sources (such as MRO or ISC requests

Transform your 3270 into a powerful SAA/CUA workstation.



CL/SUPERSESSION exploits SAA/CUA innovations for easier application integration so you can blend data from multiple applications without changing source code.

Does your 3270 make you feel like the SAA/CUA parade is passing you by... and that colorful pop-up windows, pulldown menus, and action bars are just names on a wish list?

Candle's CL/SUPERSESSION® can change all that - plus give you fast and flexible application front-ending. Or even the ability to integrate old CICS or IMS applications into new DB2 ones.

It's the *only* network productivity tool that can accomplish all this because it's the only one that's SAA/CUA-compliant. CL/SUPERSESSION's handy pop-up windows simplify even the trickiest application integration. And it supports your global network by letting you customize application front ends in different languages, including French, Italian, and German. You can also create Help screens in the appropriate language.

CL/SUPERSESSION is backed by Candle's unique Network Service Center support - a team of specialists whose mission is to make your life easier by providing fast answers to your networking questions.

CL/SUPERSESSION from Candle. Now every VTAM terminal on your network can have the zest of SAA/CUA with workstation-style productivity. For details, call us today at **(312) 954-3888** (in Illinois) or **(800) 762-7608** (outside Illinois).

"VISIT OUR BOOTH #86 AT CMG IN RENO — DEC. 11-15"

Candle®

CIRCLE #120 on Reader Service Card ▲

from other CICS regions) cannot be honored until CICS regains control. Generally speaking, compute-bound transactions, even if they retain control for as little as one hundredth of a second at a time, are somewhat incompatible with CICS' current dispatch architecture.

Estimating The Impact Of CPU Constraint

In current CICS architecture, the scheduling of tasks is essentially a single-server system; no more than one CICS task can be doing useful work in a single CICS region at a time. Borrowing a few of the most elementary principles from queuing theory, you should be able to roughly approximate the impact of CPU demand on performance. While CICS is complex enough that it will be difficult to precisely predict the effect of processing constraint, it is not too hard to estimate what level of degradation might be associated with various levels of CPU demand.

Queuing theory is simply a set of mathematical statements and formulas that can help approximate delays when you know how busy a system is. Queuing theory is equally valid estimating the delays of customers waiting in a check-out line or transactions waiting in a computer system. In the most general terms, queuing theory says that as a server becomes busier (a server is any person or thing that provides a service, such as a bank teller, a processor or a DASD device), the delays associated with receiving the service will lengthen at an ever increasing rate. The basic formula:

$$M = 1 / (1 - u)$$

provides a method of estimating the degradation for a single server based on how busy it is. In the formula, **M** is a multiplier that tells how much longer it will take to receive service than when the server is almost idle; **u** is simply the utilization (or percent busy) of the server. If, for example, a server were 50 percent busy, then, on the average, it would take twice as long to receive service as it would when the server was lightly utilized ($M = 1/(1-.50) = 2$). On the average, half of the time would be spent receiving service from the server and half of the time would be spent in a *queue* waiting while the server was being utilized by someone else. Similarly, it would take four times as long to receive service when a server was 75 percent busy ($M = 1/(1-.75) = 4$) and 10 times as long at 90 percent utilization. It would take the same amount

of time to actually receive the service, but it would take increasingly longer to get to the server as it became busier.

Since the main CICS TCB can avail itself of only a single processor at a time, CPU demand is a better measure than CPU utilization for measuring constraint in a CICS environment. Queuing delay will be underestimated unless the portion of the time that the processor is unavailable to CICS is included. Similarly, queuing delays may be overestimated in regions using VSAM subtasking or DB2 unless CPU utilization by other TCBs in the CICS region is subtracted from total CPU utilization.

When a CICS region is experiencing CPU demand of about 65 percent, you can easily see that it should take roughly three times as long for a task to receive CPU service as it would when CICS was lightly utilized. In other words, if a task actually used about 50ms of CPU, it should take about 150ms to actually receive that CPU service. Not counting increased delays in DASD or other components, queuing for CPU service would add about one-tenth of a second to transaction response time in this example.

Unfortunately, the CPU-related delays actually experienced by a CICS task may be quite a bit higher than those predicted by queuing formulas. Some factors which contribute to overall CPU demand do so in rather lengthy bursts. For example, while a page fault is being serviced, no work can be dispatched by CICS. Similarly, when a task performs lengthy internal processing (such as user-written internal sort routines), CICS cannot dispatch other tasks. When delays of this type occur, other work (such as the arrival of new transactions or the completion of I/O events upon which other tasks were waiting) will build up within CICS. When CICS eventually gets the chance to dispatch other tasks, the amount of work waiting to be dispatched will cause CPU demand to be momentarily higher than what the average might indicate. This, of course, will result in still longer CPU-related delays.

When long delays contribute a significant portion of CICS' total CPU demand, system performance will mimic a system with a higher level of CPU demand. The system will perform like a series of periods each with higher CPU demand. In other words, while occasional page-ins, CI or CA splits or other lengthy delays may not be noticeable, when they occur with enough frequency to affect the level of CPU demand, their impact will be multiplied more than indicated by queuing theory. The net effect will be an uneven level of service. Sometimes users will get acceptable response times and sometimes they will not. While the overall average response time may still appear acceptable, users will tend to remember the longer delays and may perceive that they are receiving poor service.

CPU Constraint And MRO

CPU constraint can be particularly troublesome in an MRO environment. Even though MRO has been used to break up processing and allow multiple CICS regions to share what was once a single workload, it is not uncommon to find MRO regions which are CPU constrained. Depending on the regions in which applications and files are defined, some remotely accessed resources may be resident in regions experiencing relatively high levels of CPU demand. If such resources are referenced heavily from other regions, this can have a negative impact on the performance of tasks accessing them.

When a request is made to access a file or other resource via MRO, the requesting region will communicate the request to the Resource-Owning Region (ROR) by posting information in a TCT system entry in the ROR. If the ROR were idle (and with Release 2.1, if the appropriate number of MRO requests had been received), the ROR would be activated to service the request. If, on the other hand, the ROR were already active and an application task had been dispatched, the application would continue to process and the ROR would not recognize the MRO request at that time. Later, after the application issued a CICS request requiring it to wait, the ROR would invoke its dispatching logic, recognize the MRO request and service it. Similarly, if I/O were required for the request, the ROR might not immediately recognize the completion of the I/O event. Thus, an MRO request received by a CPU constrained region could experience multiple scheduling delays.

High levels of CPU demand may not necessarily have that severe an impact on MRO service. Since mirrors usually run with a high internal priority, any MRO requests received will be serviced whenever CICS has the opportunity to do so (unless MRO batching is being used). However, if CICS tasks in the ROR are compute-intensive or if factors like paging or CI/CA splits restrain the ROR from using the CPU, the impact on MRO service can be substantial.

Overcoming CICS CPU Constraint

There are several ways to lessen the impact of CPU constraint in a CICS environment. In general, these techniques fall into four general categories:

1. Reduce CPU usage by internal CICS tuning
2. Split processing among multiple processors
3. Tune environmental factors which interfere with CICS processing
4. Prevent tasks from monopolizing processing in CICS.

Depending on the nature of the constraint, different approaches can be beneficial. If CPU utilization by the main CICS TCB is high, reducing processing or shifting processing to other TCBs can be beneficial. If CPU demand is high but CPU utilization is reasonable, adjusting environmental factors might provide more relief. If application tasks dominate the region with lengthy processing, action should be taken to address task processing.

CPU Tuning Options

Turn Off Trace

The trace facility is one of the largest consumers of CPU in CICS. It is particularly expensive in an MRO environment because of the number of trace entries generated by MRO-related modules. Typically, trace will consume from 10 to 20 percent of the cycles utilized by a CICS region. Not only is trace a major consumer of processor resources, it is not really that useful in many high-activity systems. Even with a large trace table, there is a good chance that the entries you might be looking for would have rolled out of the table before they could be printed.

Tune Your Monitors

Almost without exception, unless no monitoring is being done, the largest single user of CPU in CICS systems is the monitor collecting detailed CICS statistics. Most CICS systems use either IBM's CICS Monitor Facility (CMF) or The Monitor for CICS by Landmark Systems Corp. (Vienna, VA) to collect statistics which are used for performance analysis, reporting and capacity planning. Boole and Babbage's (Sunnyvale, CA) CICS Manager also uses CMF (or what Boole and Babbage calls CMP) with its own enhancements to collect additional information. It will perform about the same as CMF but will use a little additional CPU. There is some tuning that can be done in both the CMF and Landmark monitors to reduce the amount of CPU used, sometimes substantially.

The most expensive data to capture with either monitor is information on CPU utilizaton and paging. Both monitors must issue an MVS SVC immediately before any task is dispatched and as soon as CICS regains control. The SVC updates CPU utilization for CICS. The monitors then use this updated information to post CPU utilization for individual CICS tasks. My monitoring has shown that about half of the cycles spent in either CMF or The Monitor are spent in this SVC. Unless you actually plan to utilize this data, CPU usage can be reduced significantly by not collecting it.

Both monitors allow you to avoid collecting CPU statistics and collect only *dispatch* information. Dispatch time is the time a task had control and was, at least in theory, trying to make use of the CPU. (The monitors simply save a time-stamp before giving control to a task and another when control is returned.) Unlike the CPU data, the dispatch statistic also includes delays for paging, MVS dispatching and other overhead. In many systems, dispatch and CPU measurements will be similar. A large difference between dispatch and CPU times will give an indication of environmental degradation.

There is additional tuning that can be done in either monitor. In CMF, you can collect only those classes of information that actually will be used. Most installations use only the performance and possibly the exception classes of data. Accounting data is seldom used and that class is seldom activated. You can also tailor which performance data is collected by CMF (CICS Releases 1.7 and above) or The Monitor (Release 8.0). This can help reduce CPU consumption. In The Monitor, you can also select whether to include the optional file statistics by transaction. Collecting this data can account for as much as 20 to 30 percent of the CPU used by The Monitor. Again, if this is not information you are going to use, you may not want to collect it.

In most CICS regions, the untuned CMF or Landmark monitors will each consume from 20 to 40 percent of the total cycles used. Systems which are inactive and use little CPU will spend a larger percent of the time in monitor code. In other words, if a region were using only five percent of a processor, 35 to 40 percent of that five percent activity might occur in monitor code. When that same region was running at 65 percent CPU busy, a smaller percentage (perhaps about 25 percent) of the processing would be in monitor code. This would be true with either CMF or The Monitor. With tuning, it should be reasonable to run either monitor collecting CPU statistics and consume about 20 to 25 percent of the path length of a moderately active CICS region. If CPU usage statistics are not accumulated, it would be reasonable to reduce that percentage even further.

In CICS Release 3.1, major changes are planned for the way CMF will collect CPU statistics. Instead of having CICS continuously reissuing an MVS SVC to synchronize CPU statistics, this information will be collected for CICS tasks by the operating system (ESA). It is supposed to reduce the cost (in terms of cycles spent) of collecting CPU statistics in CICS. Other changes in internal CICS architecture should also reduce the path length required to interface with CMF.

As a final note, monitors such as OMEGAMON for CICS by Candle Corp. (Los Angeles, CA), the external component of The Monitor or the real-time portion of Boole and Babbage's CICS Manager will not operate under the main CICS TCB and should not impact CPU processing unless total CPU usage (for the entire system) was high (almost at 100 percent). These types of monitors are not designed to collect detailed statistics about specific transactions but to provide a statistical picture of CICS through sampling or an analysis of data available from an internal monitor. As a result, not only do these monitors *not* impact the main CICS TCB, they also do not usually use much CPU in accomplishing their function.

Place Files Under LSR

Local Shared Resources (LSR) provide a facility that allows a reduction in the number of I/O events required to access most types of VSAM data. When VSAM files are accessed via LSR, records will be retrieved directly from buffers if they are already in memory. This will reduce the number of times VSAM must actually schedule physical I/O events to service application requests. With Releases 2.3 and later of Data Facility Product (DFP), VSAM's buffer search algorithms have been improved to allow large LSR pools. In most cases with larger LSR pools, more records will be located in memory and CPU utilization will be reduced as fewer physical I/O events are scheduled. IBM SEs have a tool (the VSAM Large Buffer Pool Analysis Aid) that allows the analysis of current LSR pools. This utility can help you determine how many I/Os may be saved by increasing the size of LSR pools. LSR can be used to reduce CPU utilization in CICS regions by reducing the number of I/O events scheduled.

Reduce The Number Of MRO Threads

Each time CICS checks for the arrival of new MRO requests, it must scan the entire list of MRO strings defined for each region. If an excessive number of strings has been defined, it will take more processing each time the scan is done. It is good to have enough threads defined to service normal MRO traffic with few link failures. Having too few threads will cause unnecessary delays; too many threads will waste CPU cycles.

Use Long-Running Mirrors

Release 2.1 of CICS introduces an option called long-running mirrors. Normally, a mirror task will remain attached to the requesting task only as long as it is actually needed. When the long-running mirror option is used (SIT parameter MROLRM = YES), mirror tasks will remain attached to their requesting tasks until a sync-point is taken. By retaining mirror tasks, they can be reused for multiple unrelated MRO requests by the same task. This will save the processing associated with reattaching mirror tasks for subsequent MRO requests.

Long-running mirrors will be beneficial primarily in an environment in which applications issue multiple unrelated function shipping requests to the same region. Long-running mirrors will not affect browse processing since the mirror task would be held anyway as long as the browse was active. Similarly, update requests following read-update requests would still use the same mirror and update requests for logged files would, of necessity, cause the mirror to be retained until a sync-point was taken. The use of long-running mirrors will be of primary benefit in an environment in which transactions issue multiple read requests or short browse requests for files in the same remote region. If this type of processing occurs frequently, long-running mirrors can reduce total CPU utilization.

However, the use of long-running mirrors will not always have a positive effect. For one thing, in a heavy MRO environment when mirrors are held until sync-point, additional mirror tasks will need to be created for MRO requests from other transactions. Moreover, there will be some additional processing at sync-point. Depending on the number and type of MRO services required by a typical transaction, long-running mirrors may cause as much as a 20 to 25 percent *increase* in CPU consumption by mirror tasks.

In order to determine what impact long-running mirrors have in your environment, you will need to compare the *total* amount of CPU used by mirror tasks with and without this feature in similar processing periods. This number is most easily calculated by multiplying the average CPU per mirror task by the number of mirror tasks shown in CMF or The Monitor statistics. With long-running mirrors, the total task counts shown by these monitors should decrease significantly. (Both of these monitors will write a performance record for a mirror transaction whenever the mirror is detached from a transaction.) However, the average CPU recorded for each transaction should increase because of additional work being done per attach-disattach session. If total CPU usage increases with long-running mirrors, it is likely that the total number of real mirror tasks created (as shown in normal CICS shutdown statistics) will have increased. This would reflect the creation of additional mirror tasks while long-running mirrors were being held by other tasks.

Batching Mirror Processing

Release 2.1 of CICS also introduced an option called batched mirror processing (SIT parameter MROBTCH). This feature allows CICS to forego the processing necessary to recognize incoming MRO requests until the specified number of requests have been received. The use of this option will save cycles by allowing CICS to minimize the number of times it scans the TCT system entries, but it will delay the initiation of MRO services.

There are two situations in which the use of batched mirrors may be particularly troublesome. In times of light activity, it may take quite a while for enough requests to arrive to trigger MRO processing. Tasks could be delayed as long as the ICV interval at which time all requests would be serviced.

The other potentially difficult situation is RORs with high CPU demand, especially if much of that demand consists of external delays (such as paging) or lengthy internal processing. When CPU demand is high, there can be lengthy delays between the times CICS has the opportunity to recognize MRO requests. For this reason, it may be worthwhile to allow MRO-related work to occur even if only one or two requests have been received. An indication of the performance impact of this parameter can be seen in the total response time shown in monitor transaction statistics (the product of average response time and total mirror sessions) for comparable periods with batching thresholds set at different levels. Total CPU consumption can be calculated similarly. The difficulty is determining whether reductions in CPU demand improved performance enough in the ROR to compensate for the service degradation associated with batching.

If long-running mirrors and MRO batching are both being implemented, I would *strongly* recommend that they be installed and evaluated separately. Long-running mirrors may actually increase CPU utilization and MRO batching may degrade performance. If both features are installed simultaneously, it may be difficult to determine their separate effects. It also may not be possible to fairly judge the impact of MRO batching when long-running mirrors are being used since mirror tasks will always be held until sync-point.

Use CICS Data Tables With ESA

CICS Release 2.1.1 introduced a feature called data tables which can be used only in ESA environment. These tables may be either CICS or user maintained. With CICS Maintained Data Tables (CMDT), all or part of a VSAM KSDS will be loaded into storage above the line and maintained by CICS. The data in memory will be the same format as data on disk. CMDT even supports file updates for files in data tables. When the entire file is not in memory, accesses for data not in memory will be passed through to VSAM. The use of CMDT will be transparent to applications — it will appear as if they are accessing VSAM files.

The primary benefit of CMDT is that it will reduce the path length required to retrieve VSAM data. With data tables, the hashing algorithm used to locate data is, according to IBM benchmarks, faster than comparable LSR access, even when LSR achieves a similar data hit ratio. With files for which all or most active records can be loaded into memory, data tables promise to provide excellent service and a reduction in overall CPU utilization.

It would seem, though, that some caution needs to be used in choosing which files are placed in CMDT. CMDT will cause additional overhead for files which are heavily udpated — data will need to be updated in tables as well as on DASD. Additionally, when the entire file is not loaded into memory, LSR may provide better service in many cases if data access patterns vary during the day. Once data tables are loaded, the records in memory have been set; with LSR, data access patterns will determine which CIs will remain in memory. Unless an entire file can be loaded in memory or you can be quite certain that the portion of the file that is loaded is in fact the portion against which most of the activity will occur, LSR may be a better choice for both performance and CPU utilization.

Tune Temporary Storage

There are a few things that can be done to help conserve CPU cycles accessing Temporary Storage (TS). The first and easiest is to increase the number of TS buffers. In most cases, larger TS buffer pools will reduce both read and write I/Os. (TS WRITE requests to AUX are not necessarily written to external storage if enough buffers are present.) Depending on the nature of TS activity, somewhere between eight and 50 buffers should eliminate most physical I/O activity associated with auxiliary TS requests. Of course, reducing the number of physical I/Os should reduce CPU utilization. The main drawback in increasing the number of buffers is that TS buffers are allocated *below the 16MB line* and will affect the size of the DSA.

A second thing that can be done to conserve cycles is to place TS data which is to be heavily accessed (such as information loaded at startup and accessed all day) into main TS. The path length to access main TS is much shorter than that to access auxiliary TS. However, since main TS is above the line, in releases of CICS prior to 3.1, MVS getmains and freemains will be required whenever elements are created or deleted. Small data elements which are frequently created and deleted can cause fragmentation of MVS storage above the line and may cause main TS processing to become more CPU-intensive than equivalent auxiliary functions. It is also worth considering that since main TS is above the line, large users of main TS will increase the working set size of the CICS region.

Splitting Processing Among Multiple Processors

VSAM Subtasking

VSAM Subtasking Program (VSP) is an easy way to off-load some processing from the main CICS TCB. When VSP is being used, VSAM, TS and transient data file processing will be handled by the VSAM subtask. Shifting this processing will lower CPU demand in the main CICS task, but at a cost of increased overall CPU usage. Since VSP overhead can be substantial, it is usually recommended that VSP not be used unless CPU demand exceeds 60 to 70 percent.

With CICS Release 2.1.1, VSP has become more efficient and less resource intensive. VSAM browse requests will never be sent to the subtask (they require little processing and cause relatively few real I/O requests). VSAM update requests will always be processed by the subtask (update requests always result in requests to write data). VSAM read requests will be processed by the subtask only if the main CICS TCB is experiencing CPU demand more than 70 percent. Under 2.1.1, you can obtain most of the benefits of VSP without as much of the overhead.

Use of MRO — Multiple CICS Regions

Although MRO was initially viewed primarily as a means of relieving virtual storage constraint, it also has become an important means of relieving CPU constraint. Depending on the nature of the application systems involved, the complexity and cost (in terms of additional CPU usage) of creating additional regions will vary immensely. If applications are relatively self-contained sharing few resources with other systems, it is often a relatively minor task to move one or more systems and all of their resources (files, TS and so on) to a new Application Owning Region (AOR). Moves of this type tend to be almost free, especially if messages had already been routed through a Terminal Owning Region (TOR). CPU demand would be decreased in the original AOR and total CPU usage would increase only slightly. In this case, the only significant cost would be the increase in working set necessary to support a new CICS region.

When application systems cannot be divided cleanly, the creation of new MRO regions will probably introduce overhead and increase total CPU utilization. Several difficult choices may need to be made when splitting closely interrelated applications. It is normally wise to define files in the region that will access them most frequently. This will save the overhead associated with function shipping requests for file access. However, if the region owning the files is itself CPU constrained or has tasks which are compute-intensive, this may present a bottleneck for other regions accessing resources remotely. Depending on the number of accesses and level of CPU demand, it might be beneficial to define files in other AORs or create File Owning Regions (FORs).

The advantage of defining resources in an FOR is that the FOR should experience low CPU demand and provide quick access to files. The main drawback to having an FOR is the overhead associated with shipping all file accesses to another region. An FOR should be able to provide service for most types of files almost comparable to local access but will increase total CPU utilization. However, files which are heavily browsed would probably be poor candidates for an FOR. The processing associated with function shipping is so much greater than that associated with typical browse operations that the costs of moving a heavily browsed file to an FOR may be quite noticeable.

Of course, the major goal in splitting CPU-constrained regions is to reduce CPU demand in each of the new regions. Enough processing must be shifted to materially reduce peak CPU demand. When planning an FOR it is worthwhile to look at applications contributing processing during peak periods. It does little good to split applications into separate regions unless their processing peaks overlap.

Improving Environmental Factors

Paging

During the past decade paging has been one of the most serious detriments to CICS performance. Few things could degrade CICS performance faster than an excessive paging rate. Generally speaking, when paging to standard DASD devices, page rates more than two page-ins per second were considered excessive because of their effect on CPU demand. Any number of things have been done to mitigate the degradation associated with paging. Faster I/O devices (such as drums, cache controllers or solid state disk) have been used to reduce page service time. Storage isolation has been used to protect the CICS working set. Various other tuning has been done to reduce page service time.

In a CICS environment, though, the best solution is to have enough processor storage that paging does not occur. Particularly in a CPU constrained environment, it is important to have enough real or expanded storage that CICS does not wait for page faults to be serviced from external DASD. While real storage will provide better service than expanded storage, in most cases, expanded storage will provide adequate service. Even though a certain amount of overhead is expended satisfying page faults from expanded storage, unless the page fault rate is relatively high, the difference between adding real or expanded storage may not be noticeable. Only when total CPU usage is high and the page-fault rate is significant will it make much difference whether real or expanded storage is added to eliminate paging.

Dispatching Priority

CICS is typically favored in most operating environments. Generally, CICS is given a higher dispatching priority than all other workloads except JES, VTAM and the operating system. In most cases this is acceptable. With multiple CICS regions, service regions such as FORs and TORs should normally be given higher priorities than other CICS regions. Beyond that, the regions which use the most CPU should be given the highest priority, especially if they are CPU constrained and running in a multi-processor CEC. If a constrained region also serves as a significant ROR for other regions, it is important to favor that region as much as possible, perhaps even to the point of raising its priority above JES and VTAM.

It is important to remember that even though batch work is given a lower dispatch priority than CICS and will only use *left-over* cycles, much of the I/O processing is performed by the supervisor at the highest dispatching priority. When a CEC is constrained, it is not wise to run a lot of batch work during peak periods since some of that processing will interfere with CICS' ability to access processors.

ESA

When CICS is running on 3090 S-class processors, it will perform better with ESA than XA, particularly in CECs with five or six processors. ESA has code designed to redispatch CICS regions on the same processor allowing more efficient internal processor caching. The net effect is that CICS systems running on a 3090 Mod 600 S should receive the equivalent of five percent more power running under ESA than under XA, even if nothing further is done to exploit ESA.

Through the use of hiperspaces for VSAM buffers, ESA can further leverage performance and reduce CPU utilization. When VSAM buffers are placed in hiperspaces, they may be accessed more efficiently than simply mapping them into expanded storage since hiperspaces may be accessed directly without experiencing page faults. While placing buffers in hiperspaces may not directly improve LSR performance, it can reduce CPU utilization by trimming CICS' working set and reducing the number of page faults satisfied from expanded storage. Of course, the amount of benefit this can provide will vary with the number of page faults per second being satisfied from expanded storage.

Preventing CPU Monopolization

Move Compute-Intensive Code To A Separate CICS Region

Perhaps the best thing to do with compute-intensive code is to tune it or get it out of CICS. Assuming that code is tuned and must be run under CICS, the next best option is to place it in a separate CICS region by itself. Transactions which are notoriously compute-intensive will do less harm when they compete only with other similar tasks and do not obstruct typical CICS workloads. MVS parameters can then be used to control which CICS regions are to be favored. The only restriction in creating special processing regions is to ensure that they do not contain any files or resources that can be accessed via MRO. MRO access to processing-intense regions may receive unacceptable service.

Issue CICS Suspend Command To Break Up Processing

Programs can issue a suspend command to allow other higher priority CICS tasks to process. If heavy compute modules can periodically issue the suspend command, other tasks with higher priority will be allowed to process. While this sounds good in theory, it is often not that simple. For one thing, unless special monitors are available, it is not always readily apparent where processing is occurring. Furthermore, once processing concentrations have been located, it can still be difficult to determine where and with what frequency to execute the suspend commands. And doing all this, the transaction must be unique and eligible to run with a lower CICS internal priority.

Eliminate CI And CA Splits

Prior to Release 2.1.1, VSAM CI and CA splits would cause the entire MVS task to be suspended while split processing was occurring. This would mean that either the main CICS TCB or the VSAM subtask (if VSP were being used) would be suspended until split processing completed. With Release 2.1.1, CICS uses a new VSAM exit to relieve this situation. This should allow CICS to continue to process while a split is taking place asynchronously.

Other Considerations

DB2

DB2 will normally require considerably more total resources to access data than would be used by native VSAM. Replacing VSAM files with DB2 tables will increase total CPU utilization. However, DB2's architecture should make better use of the processor, especially in a multiprocessor environment.

Unlike other workloads under MVS, DB2 has been designed to exploit n-way processors. It used numerous MVS tasks to pass data and communicate. Thus, while it is more resource intensive, DB2 can still provide acceptable service at higher levels of CPU utilization. Unlike CICS single TCB architecture, which becomes constrained around 60 percent CPU demand, DB2 should provide reasonable service even when total processor utilization is high. In a recent IBM briefing I attended at Poughkeepsie, NY, it was stated that IBM has special code in S-class processors to allow DB2 to exploit n-way processors and that future processors would have features that would further enhance DB2's multi-tasking architecture. It appears that IBM will be incorporating DB2-related performance enhancements in future processors.

CICS Release 3.1

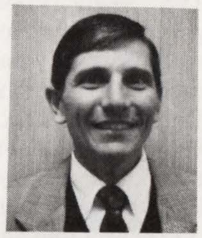
An examination of CICS Release 3.1 gives the appearance that IBM is trying to position CICS to dispatch multiple workloads in a single region. Major efforts have been made to segregate processing into multiple functional areas. With the demise of macro-level support in some release after 3.1 and with CICS' internal restructuring, I believe that eventually CICS will allow application dispatching from multiple TCBs in a single region. If and when this happens, CICS will be able to make better use of multiple processor environments. At that time, most of the constraints associated with CPU demand will vanish. Unfortunately, it may be years before seeing such an architecture in any release of CICS. It cannot happen any earlier than the release after 3.1 when macro-level support is finally dropped.

Conclusions

CICS has come a long way in the past few years. IBM has done a reasonably good job of enhancing CICS functionality and performance. Most of the more common performance constraints have been solved through numerous product enhancements. It appears that CPU constraint is now one of the few remaining architectural impediments to CICS performance.

CPU demand is the best indicator of potential CPU constraint. It consists of the time that CICS is using or trying to use a processor. When CPU demand is high, CICS performance will suffer. Various techniques can be used to lower CPU demand including tuning within CICS to lessen CPU usage, splitting processing with VSP or MRO, tuning the MVS environment and breaking up processor intensive routines in programs. CICS' internal dispatching structure may restrict application performance, but there are many things that can be done to reduce CPU demand and overcome performance limitations.

ABOUT THE AUTHOR



Ted C. Keller is the manager of a group responsible for CICS systems support, performance management and capacity planning at Yellow Freight System, Inc., 10990 Roe Ave., Overland Park, KS 66211, (913) 345-3274. He has worked in various data processing jobs for more than 21 years.

VM Data Communications

By Ed Sterling

It has been said that VM was killed many times by IBM, yet today VM is stronger than ever and has taken an equal place beside MVS. Many people see the two operating systems as black and white, wanting one to be better than the other. In reality, each operating system does a good job in a certain major area and VM is best known for its interactive timesharing support. The mention of timesharing immediately suggests users on terminals, wanting to connect into and out of a VM system on many different media. It also means users wanting to exchange data with one another, between large systems, minicomputers and personal computers. It is the connectivity game and it is increasingly complex. Yet in some ways, it is a nice problem to have. There are so many choices it is difficult to know which is the best one.

In the early years of VM during the 1970s, data communications support was primitive. Card image files could be transferred between hosts, a long-time IBM traditional communications path. Print files could be sent to remote printers. Users on 2741 "Selectric" typewriter terminals as well as ASCII terminal users could logon in "line by line" (teletypewriter mode). In the first release of VM/370 (circa 1972), there was no 3270 suppport!

By 1980, IBM realized VM could not be killed and moved to make it a "strategic" product. Today, VM's data communications capabilities are certainly on par with MVS, especially since SNA/VTAM is available in both operating systems. The importance of the VM system as a premier application development platform and as a migration vehicle for new releases of CICS, IMS and even MVS requires the full scope of communications support used by these subsystems. And because the VM environment is so conducive to development and so much control can be given to the developer without compromising system integrity, VM becomes an obvious choice for data communications research and development.

VM Data Communications: Past And Present

VM Almost Killed By IBM

VM is nearing the end of the second decade of its commercial existence. During the "early years" of the 1970s, VM/370 (as it was then called) was often run on a separate machine as something of an outcast or a non-conformist. If the MVS group regarded the VM system with disdain, the feeling was usually mutual with the VM programmers delighting in their new-found extended control and improved interactive response time. As a result, VM systems were usually not part of the corporate mainstream network; thus, simple communications were generally regarded as adequate. In fact, because VM was regarded in many ways as experimental, even inside IBM, many installations spent their days making (often extensive) modifications and debugging problems both for themselves, other installations and IBM. Networking was generally minimal.

Early VM Networking: RSCS

The earliest networking package available for VM was (and still is) RSCS, the Remote Spooling Communications Subsystem, introduced as an integral part of VM/370 Release 2 in 1974. RSCS might be thought of as a simple VM version of HASP, the time-honored remote spooling system from the 1960s. It consisted of a virtual machine with its own operating system; this was done primarily to simulate multi-tasking, which VM does not support in a single virtual machine. RSCS could communicate with a remote VM system, a HASP system or a non-programmable 2780 workstation that performed remote printing. The links used Binary Synchronous Communication ("bisync" or BSC) and while each link was operational, the line was polled every 2.5 seconds to ensure the other side was operational. However, there was an inherent "master/slave relationship" between the sender and the receiver, instead of a more logical peer-to-peer especially between two VM systems. Another major problem was that all communications were point-to-point: there was no ability to pass files on to a more distant third system using network routing. But, for simple remote printing and exchange of punched-card images, RSCS cost nothing and did an acceptable job.

As VM grew in popularity and VM installations began to grow internally, the need for some RSCS networking upgrades became evident. The idea of logging into a remote VM system from your own terminal was still the "stuff dreams are made of." However, the primary need of the day was for "store and forward" networking, allowing the establishment of a true "network web" of processors and each VM system to see each other as equals. Better performance and lower overhead were also becoming important.

Various short-lived experimental systems appeared to fill the gap. Several large VM installations decided to "go their own way" and created an informal consortium called the Common System. Besides sharing VM modifications with each other, they designed and built a significant networking system called RASP (something of a merge between RSCS and HASP) based on the then-current RSCS. RASP addressed many of the concerns VM installations had voiced at the SHARE user's group but which had not yet been addressed by IBM. RASP ran successfully at about a dozen installations in the late 1970s.

VM Becomes A Strategic Product

To be sure, a lot was going on inside IBM concerning VM during the mid-to-late 1970s. IBM internal interest in VM was growing sharply and IBM's own internal Subsystem Unified Network (SUN) was based on the IBM internal RSCS. To distinguish it from the regular RSCS, this internal version of RSCS was named VNET. Started in 1972, VNET continued to be developed and was finally made available to VM customers as a PRPQ (that is, special controlled release) in 1976.

VNET addressed many of the SHARE requirements: it offered peer-to-peer networking, low overhead during idle periods, full CMS file support, store-and-forward networking, Channel-To-Channel Adapter (CTCA) support for linking VM to other systems in the same physical complex and remote 3270 workstation printing. Yet despite these extended capabilities that VNET offered, its price was simply too high for most companies to justify and its use at customer sites was never significant. Finally, as VM's status improved and IBM moved to create VM/SP from VM/370, VNET was also revised for a regular Program Product release and became available in 1979 as RSCS/Program Product (RSCS/PP) at a reasonable monthly cost. The old RSCS, RASP and VNET quickly disappeared as everyone adopted the new Program Product as the "standard."

VM Networking Extends To Terminals

Concurrent with the creation of VM/SP came another long-sought communications offering: the ability to logon to a remote VM system as if it were part of your local CPU. This was achieved using an elegantly simple subsystem added to VM's Control Program (CP) called Logical Device Support Facility (LDSF) and a new networking product called VM/PassThrough Facility (VM/PASSTHRU). VM/PASSTHRU became an essential tool in VM system maintenance with the introduction of the tiny 4331 processor. Being one of the first "departmental" processors in the mainframe class, the 4331 required mainframe system programmers, yet the cost could not be justified. VM/PASSTHRU was the solution because it brought the remote VM 4331 system right to the system programmer's terminal. Now, in conjunction with an RSCS link to send fixes, modifications and maintenance procedures to one or more of these remote systems, SYSGENs could be performed without traveling to another terminal, dialing-up remotely or even physically going to the remote site for such maintenance. The combination was superb and paved the way for effective remote-site maintenance from a central location. Like the new RSCS/PP, VM/PASSTHRU treated VM systems as peers, used either BSC or CTCA links between systems and was reasonably priced.

The First Attempt At SNA For VM

Reeling from the delight of IBM's "blessing" VM in the form of VM/SP and the advent of RSCS/PP and VM/PASSTHRU providing low overhead high-function networking, VM installations' networking needs seemed well addressed by 1980. Yet one could not ignore IBM's true corporate networking strategy: SNA. The spectre of two separate corporate networks loomed large: SNA for the MVS systems and RSCS and PASSTHRU for the VM systems. Clearly, while this might have delighted the equally bigoted systems programming groups, neither IBM nor its major customers could tolerate this. VM was winning its case as a highly effective development platform and the success of the PROFS electronic mail system was selling VM processors by itself. Somehow, VM had to link into the corporate SNA network.

IBM's quick answer was to move the MVS version of VTAM into a virtual machine and have it controlled by a "guest" operating system. OS/VS1 was the obvious choice since it had a special "VM handshaking" feature to reduce duplication of effort in paging. From IBM's standpoint, this new package called VM/VCNA was an easy way to address the SNA link-up problem. VCNA used mainly off-the-shelf components and the only major change to VM was the addition of IUCV Console Communication Services (CCS). CCS seemed to duplicate the Logical Device Support in some respects, although it allowed for a more general concept of remote logons (3270 and ASCII) for remote SNA users.

In practice, VCNA was a square peg in a round hole. Early users had a difficult time installing and running it and many "VM purists" would have nothing to do with a VS1 system running on their CPU. The 1976 design specification for VCNA could not foresee the popularity of VM and many CMS-intensive installations that would exist three years later. Most installations regarded the overhead of running a VS1 guest system as too high a cost to simply link terminals to the SNA network. VS1 expertise was required, as well as a knowledge of traditional OS, JCL and so on. The requirement still stood for a "native" SNA implementation with VM-like installation procedures and maintenance tools.

VM Goes Native With SNA Networking

Finally with the advent of VM/SP Release 4, IBM announced native VM/SNA support. A new "operating system" was provided by IBM in VM/SP called the Group Control System (GCS). GCS is like a miniature version of MVS whose sole function is to provide all the required OS services to make ACF/VTAM operate in a virtual machine. Major changes were made in the VM CP to allow for the emulation of MVS' CSA, so that common storage can be shared by members of the group running the GCS operating system. A new version of VTAM for VM was developed and is required for the operation of VM/SNA; it forms a custom fit with the GCS to provide SNA networking. All maintenance procedures and files are done using native CMS-like files and EXEC procedures. VM/SNA effectively replaces VM/VCNA; only installations still running VS1 for production might still wish to run VCNA.

Thus, VM/SP now has a full complement of networking products to satisfy both the VM "purist" as well as the MIS director who needs full connectivity to the corporate network.

VM/SP Support For Data Communications Hardware

You can attach any kind of device to a VM system. The question becomes whether CP has support for it or you have a device driver in a virtual machine that can control the device. It is, therefore, pertinent to review which devices are supported directly by CP. SNA support is detailed in the next section under IBM Data Communications Products: VM/VTAM.

37x5 Front-End Communications Processor Support

The 37x5 (3705, 3720, 3725 and 3745) front-end processors provide sophisticated communications support for SNA, non-SNA and certain Local Area Network (LAN) connections. Traditionally, VM's CP has had support for the 37x5 Emulator Program (EP), as well as real 2703 type transmission control units (the few of these that exist today are usually DEC PDP-11s emulating the 2703 or the venerable Memorex 1270). EP supports asynchronous (ASCII) terminals and bisync (BSC) lines for RSCS and remote 3270 terminal clusters. When running EP, the 37x5 is "owned" by the VM CP as opposed to a virtual machine such as VTAM. The EP is stored in a saved segment (DCSS) and the CP NET LOAD command is issued by the operator after IPL to load the 37x5 with a specified version of the EP. For SNA operation, ACF/

MULTIPRINT/VM Does More Than Deliver Printed Output...

Just

It also delivers flexibility in an easy to use product.

Are you tired of using a print routing product that's difficult to use by operations personnel, too basic for your current needs, poorly supported, or won't let you print on a VTAM printer from CMS? Then you and your operations staff should consider MULTIPRINT/VM™.

MULTIPRINT/VM is a comprehensive VM-based print routing product for VM as well as guest operating systems. It is the first print routing product to also support VM/VTAM with a true GCS interface. And its command format is based on the VM command set so it's easy to learn and use!

Backed by BlueLine's superior technical support, MULTIPRINT/VM provides:

- *Printing on a VTAM printer from CMS*
- *"Hot" printer support*
- *Automatic forms control*
- *Graphic printing*
- *Printer pooling*
- *Forward and backward printer spacing*
- *Support for local channel-attached printers and printers defined to VTAM*
- *Printer sharing between CICS and CMS*
- *And much more!*

*So, start routing your print with a product that does more than just deliver printed output. Call us today at **800-826-0313 (612-542-1072 in Minnesota)** and discover for yourself the difference of using MULTIPRINT/VM!*

BlueLine™
SOFTWARE INC.

New Advances in Performance, Productivity and Planning.

5775 Wayzata Blvd., Suite 690
Minneapolis, Minnesota 55416
800-826-0313 • 612-542-1072 in Minnesota

CIRCLE #48 on Reader Service Card ▲

NCP is required and because CP does not support this itself, the 37x5 must be loaded and controlled by either the VCNA or VM/VTAM virtual machine; the CP NET LOAD command does not apply under SNA.

3270-Family Display Station Support

CP supports both local and remote BSC 3274 controllers with full device support including extended highlighting, colors and GDDM graphics. A Model D local controller is channel-attached and is normally defined on a block multiplexor channel. A Model B or C remote 3274 controller attaches to the 37x5 via a BSC line under the direct control of CP. The speed of the line concerns only the 37x5 controller and not CP itself. An SNA Model A remote 3274 runs under SDLC line control and is controlled by a 37x5 running ACF/NCP.

Virtual machines run in one of two modes of screen display: a traditional *VM* mode, controlled by CP with the last two rows of the screen serving as a single command-line input area; and a *full-screen* mode, defined and solely controlled by an application program such as CICS under VM or a text editor like XEDIT. To simplify the interface to the real 3270 terminal, generic Channel Command Words (CCWs) are defined for the virtual screen I/O operations, which are later translated into the correct CCWs for the actual terminal being used. Control of a 3270 terminal can be given to a virtual machine in several ways: permanently via a directory entry linking it to a certain virtual machine or temporarily via an OPERATOR-level CP ATTACH command or the self-initialed CP DIAL command.

For non-SNA operation, a single group of remote 3270 terminals controlled by a 3274 is termed a *cluster* and there may be a total of 256 remote clusters. The maximum number of terminals supported in a single cluster is 32. Note too that *multidrop* configurations, where several controllers share the same telephone line, is not supported. This is a somewhat arbitrary decision made long ago, probably because it meant less work. CP takes the responsibility for the required BSC polling of each *cluster* of terminals; therefore, the active use of many clusters imposes some overhead on CP. Another small criticism lies in the design of the CP code for remote 3270 support causing each full-screen input to be processed twice: the initial input is effectively disregarded, while CP scrambles to switch the smaller input buffer to a larger one. In many cases, this is totally unnecessary. Two alternatives exist: switch to VM/VTAM and use SDLC protocol (a sizable undertaking) or use the remote 3270 support introduced in VM/PASSTHRU Release 3. There are also cases in which the entire VM system will ABEND (crash) due to a sequencing error in remote 3270 flow. While you can argue that this identifies an integrity problem, the entire user population suffers from a problem on a single line. In summary, having a virtual machine such as PASSTHRU be responsible for remote 3270 communications appears to be a more logical approach to VM architecture. For SNA operation, most of the above limitations do not exist. SNA supports a large number of remote 3270 controllers and allows multidrop connections.

Asynchronous Communications Support

CP has always supported asynchronous ASCII communications. However, the support is generally regarded as minimal and notably self-serving especially in the face of a large "real world" of many types of ASCII terminals. This is not isolated to VM alone, but it can be said for ASCII support in most IBM communications products. Fortunately, there are various hardware and software add-on products which make ASCII terminals function capably in a VM environment. ASCII communications support exists both in native CP and VM/VTAM.

ASCII terminals normally run in *line-by-line* mode where output appears at the bottom of the screen and scrolls off the top of the screen. A *dot* (period) character is the normal prompt, which includes an XON character to enable any terminal expecting XON/XOFF flow control. Support for this mode of input/output is generally quite limited in applicaitons which normally run in full-screen mode. In recent years, IBM has made several enhancements to VM/SP to improve scrolling and PF key usage, but it applies only to its own 3101 ASCII terminal. Hence, there is widespread use of ASCII-to-3270 protocol conversion products, which make the ASCII terminal appear as a regular 3278 terminal. This removes the "problem" of limited ASCII terminal support once and for all.

Asynchronous communication can also link a VM system with an X.25 packet switched network such as TELENET or TYMNET. IBM offers this capability only through VM/VTAM, but COMMPRO Associates (Redondo Beach, CA) sells a well-known enhancement to the 37x5 Emulator Program that creates multiple virtual asynchronous lines from a single X.25 connection. This can be a cost-effective alternative to VM/VTAM if no other SNA capabilities are required.

As mentioned above, ASCII-to-3270 protocol converters are usually the best vehicles for productive access to VM applications. There are two types of protocol converters which can be installed: hardware and software. IBM's major protocol converter is the 7171: it supports up to 64 concurrent users on many types of ASCII terminals and the unit is defined to VM/SP as a local 3274 channel-attached controller. An optional SYSGEN keyword called EMUL3270 can distinguish the 7171 from a real 3274; this allows an application program to reference a control block to determine if the terminal is really connected to a 7171. This information may be useful for extra security checking or enabling a file transfer program to a PC attached to the 7171.

Because the 7171 physically appears to be an IBM PC with a large expansion cabinet, it can run standard PC DOS software; thus, 7171 maintenance is performed using PC software. A set of approximately 12 brands of ASCII terminals is included with the system and PC-based software is included allowing the customer to modify ASCII keyboard mapping to 3270 functions or define new terminal types as required. The 7171 also recognizes a special transparency string that disables 3270 emulation and allows "raw ASCII" characters to be sent directly to the terminal or PC to drive plotters, ASCII printers or perform PC file transfer. The transparency string, an invalid 3270 addressing command, is placed at the head of a 3270 output command buffer and the "raw ASCII" follows behind it. For example, to send "raw ASCII" to the terminal, the 3270 output sequence would be EBCDIC 05C3115D7F110000 followed by the (already translated) ASCII data. Upon seeing the 115D7F110000 string, the 7171 stops all EBCDIC to ASCII translation as well as 3270 protocol conversion (but just for that output data stream on that terminal). A second code EBCDIC 115D7F110001 performs this same function and allows "raw ASCII" input from the terminal. This code is most often used for PC file transfers, allowing the PC to send ASCII file data up to the mainframe transparently.

Earlier IBM attempts at protocol conversion include the venerable Series/1 minicomputer, running the "Yale 3270 Emulation IUP," and a Series/1 variant called the 4994 Host-Loaded Protocol Converter (both packages are no longer marketed). In an SNA environment, IBM also offers the 3708 that attaches directly to the 37x5. With few exceptions, most non-IBM units emulate 3274 remote BSC or SNA controllers and require a 37x5 front-end processor. Because this setup entails two separate communications links (ASCII and then BSC or SDLC), response can be sluggish and a noticeable echo-delay may occur. However, for a group of users at a remote location, using inexpensive ASCII terminals with a protocol converter can be a cost-effective way to link to a headquarters mainframe.

Local Area Network Controllers

VM supports two LAN interface controllers: the 8232 LAN Channel Station and the recently announced 3172 Interconnect Controller (to be available next year). Both offer a high speed direct link between multiple LANS and up to two IBM mainframe channels. Both are rack-mountable and are designed for use in industrial environments.

LANs can also host attach via the 3174 Enterprise Controller and the 3745 Communication Controller. An alternative to these controllers is to have a LAN with a gateway PC on which you would run the IBM 3270 Workstation Gateway Program. PCs on the LAN would request a session with the host and the gateway PC would attach to the 37x5 as a remote 3274 controller. However, in this case, the speed of the gateway would not exceed 19.2 KBS, which is not practical for high-volume transactions.

Both the 8232 support the IBM Token-Ring LAN and Ethernet networks and both run TCP/IP as well as Manufacturing Automated Protocols (MAP) (the 8232 supports MAP 2.1 and the 3172 supports MAP 3.0). MAP is a standard designed for multi-vendor applications in the manufacturing environment. TCP/IP is a widely supported protocol, especially in the Ethernet world, and is one of the few protocols which has been widely accepted across the industry for LANs. VM runs in the host and not directly on either the 8232 or the 3172. Instead, these controllers are dedicated to the TCP/IP virtual machine that performs all initialization, I/O operations and error handling.

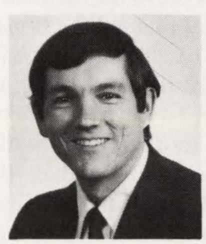
Special Notes For The 4331, 4361 And 9370

The 4331 processor (now obsolete) and its replacement, the 4361, as well as the 9370 employ integrated communications controllers to reduce space, power-consumption and cost. These units are known as the Display Printer Adapter (DPA), the Integrated Communications Adapter (ICA) and the Work Station Adapter (WSA). The DPA/WSA are essentially built-in 3274 controllers and the ICA is a built-in 37x5 communications controller. The DPA provides basic System/370 operator console support, system printer support (for the 326x family of low-speed printers) and a small base of additional 3278 ports. The ICA provides telecommunications support for up to eight ASCII, BSC or SDLC lines, running either microcoded EP or a subset of VTAM called VTAM/E. The WSA allows up to four 3299 multiplexor units, each of which expands a WSA 3270 port to eight ports. The advantages to these integrated adapters are obvious: no external controllers are required, reducing overall system costs and, in particular, keeping valuable system channel slots open for other equipment, such as disks and tape drives.

The 9370 takes a giant technological leap ahead of the 4361 by providing virtually all of the data communications capabilities of the external controllers, using its own internal subsystem controllers. These subsystems function as scaled-down 37x5, 3274, 7171 and 8232 controllers, providing full RSCS, PASSTHRU and ASCII communications, as well as a VTAM, X.25, 3270 and LAN support. Although the 9370 has been marketed as a *departmental* distributed computing mainframe, it may, in fact, become a new breed of "front-end controller," replacing a large number of different controllers with a single integrated system providing full networking and connectivity. Therefore, the 9370 may become the "gateway" to the mainframe.

More about VM data communications will appear in upcoming issues.

ABOUT THE AUTHOR



Ed Sterling is a principal founder and past president of Simware, a leading data communications software development company. He is currently developing protocol conversion software for Simware products in the TCP/IP environment. Simware is located at 20 Colonnade Road, Ottawa, Ontario, Canada, (613) 727-1779.

The ISPF Productivity Update

By Jon E. Pearkins

More Ideas For Getting Your Work Done Faster

PF keys offer an excellent method of productivity improvement by markedly reducing keystrokes. The trick is to pick the key sequences you use most frequently and then try to do as much as possible in one keystroke. One reader shares what he has done.

Program Function Keys

Assuming you have a 3270 terminal for which all 24 PF keys are accessible without having to use the Alt or Shift key, you may wish to change their meaning from the default of having PF13 to PF24 echoing the meaning of PF1 through PF12. Tom Zirtzlaff, project leader for P.A. Bergner & Co. (Milwaukee, WI), has given the matter a lot of thought and the ultimate test — the test of time.

He explains that users have the option to redefine all of the PF keys for their own use. If it is decided that this will be done, remember that keys 13 through 24 will become one through 12 if you ever logon to a terminal that has a 12-key restriction. This means that 13 through 24 should probably contain those functions that you can never do without (see Figure 1). Keys one through 12 then become the keys that you can use creatively.

Zirtzlaff has found that in the course of editing programs and text there are a number of commands he uses often. Some of these are *primary commands* such as UP MAX and some are *line commands* such as Copy and Move. Figure 1 contains his current PF key settings for ISPF. They can be changed by going to Option 0.3 of ISPF. The tutorial is helpful once you are there. Those commands that are in capital letters are the remaining original defaults (as defined by IBM). The commands in lower case have all been set up by him. Line commands are differentiated from primary commands by the colon (":") that is found in the first position of the command.

The primary commands, of course, allow you to execute those functions with one keystroke instead of several. Additionally, you do not need to "home" your cursor to the command line in order to execute the PF key. The cursor may be anywhere on the screen. Any changes that have been made to the screen are registered first, then any line commands that can be done are completed before the primary command is executed. To a limited extent, PF keys can also stack commands. The limitation is the number of characters a PF key definition can hold. The PF key 11 is equivalent to the primary command "=S.H." His shop has IBM's Spool Display and Search Facility (SDSF). PF key 11 allows you to go to a list of jobs in the print hold queue with only one keystroke.

The PF keys with line command definitions allow you to execute the line commands without the necessity of placing the cursor in the line sequence field. To enter a line command on a line, it is only necessary to place the cursor anywhere on that line and press the appropriate PF key. The line command will then be placed in the sequence field as if you had moved the cursor there and entered it yourself. This, more than anything else, saves time when editing any material.

The line commands he has put in his PF keys are single and block Copy and Move, along with the After and Before targets for them. He also makes use of the Repeat and Insert line commands and the Text Split (TS) command. The Delete line command is one that he will not put into a PF key because it would become too easy to accidentally delete lines.

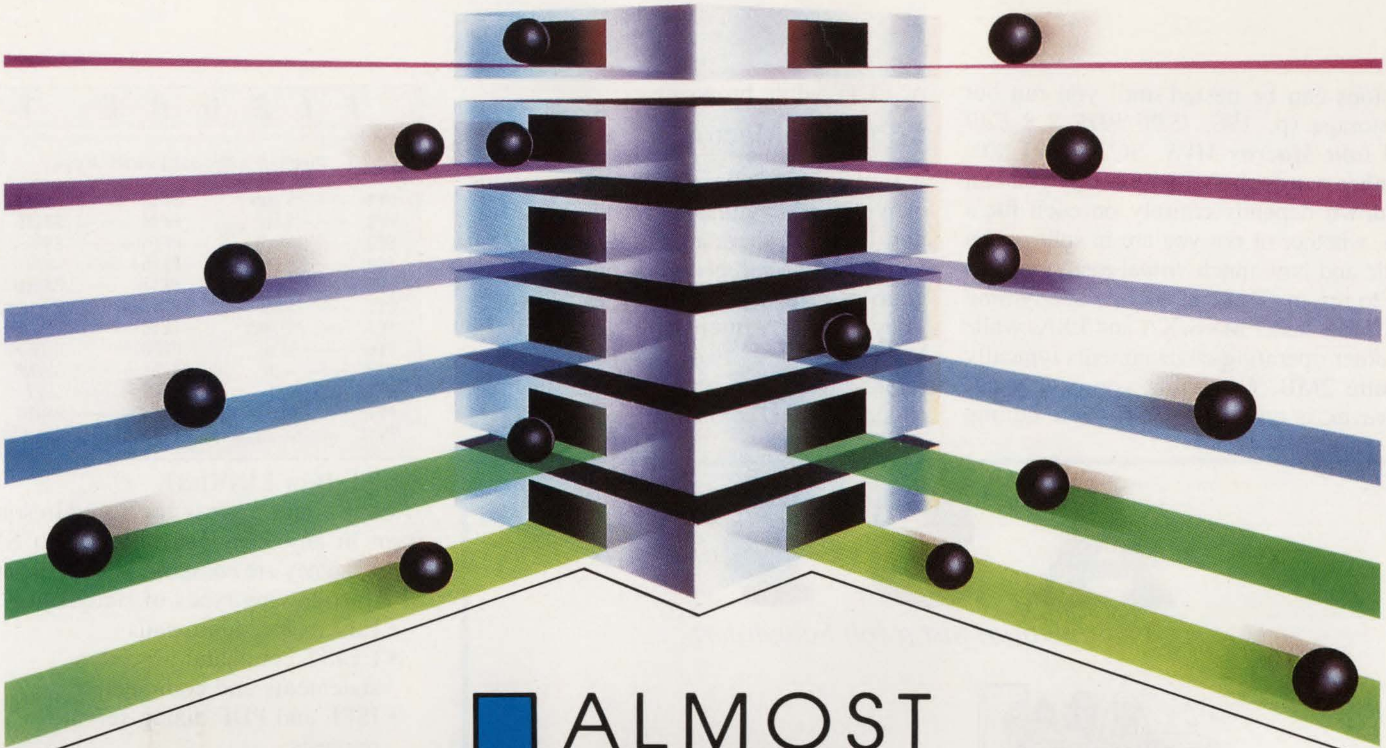
Blank plasticized templates can be used to record your personal settings for PF keys in ISPF to help you remember them when your mind goes blank or when you first start to use them. IBM even provides blank templates with most new 3270 terminals.

Recursion

Although not a favorite topic among university students, ISPF does offer one type of recursion that is both productive and easy to understand. You can issue an EDIT primary command as a method of editing a second file without using the Split Screen facility or *committing* the changes made to the first file. David Levine, data administration analyst for Sony Corporation of America, points out that you can issue a BROWSE primary command to browse a second file without getting out of the browse for the original file. Hit PF3 and you are right back where you left off in the first file.

Both commands can be used in two ways. Just typing EDIT or BROWSE will display the standard EDIT or BROWSE panel you see when you type two or one on the ISPF Main Menu. That allows you to select any ISPF Library or PDS member, as well as any sequential dataset. If the member you want to edit or browse is in the same ISPF Library or PDS, type EDIT or BROWSE followed by the member name.

Levine also points out that you do not have to stop at the second file. From the second file, you can edit or browse a third and so on. As you hit PF3 each time, you will be right where you left off in each file. Levine notes that there seems to be no limit to the number of datasets/members that can be edited. IBM is a little more precise when it states that editing



***CIRCLE #71 on Reader Service Card* ▲**

sessions can be nested until you run out of storage (p. 196, *ISPF/PDF 2.3 Edit and Edit Macros MVS,* SC34-4121-00). In other words, how many levels you can go down depends entirely on each file's size, whether or not you are in split-screen mode and how much virtual memory ISPF has to run in. You can do almost anything in 1.5MB under MVS/XA and ESA, while all other operating environments typically require 2MB. One thing you cannot do, however, is type BROWSE while editing or EDIT while browsing.

ISPF Edit Macros

Edit macros allow you to create your own primary commands for the ISPF Editor. In MVS, there are two types: CLIST *edit* macros and *program* macros. Program macros are ISPF program dialogs. They can be written in PL/1, COBOL, VS FORTRAN, Pascal, APL2 and Assembler and must be compiled and link-edited as load modules into ISPLLIB, STEPLIB or LINKLST.

FIGURE 1

Zirtzlaff's Productive PF Keys

PF1	——> sub	PF13	——> up max
PF2	——> :ts	PF14	——> SPLIT
PF3	——> :r	PF15	——> END
PF4	——> :c	PF16	——> save
PF5	——> :cc	PF17	——> RFIND
PF6	——> :m	PF18	——> RCHANGE
PF7	——> :mm	PF19	——> UP
PF8	——> :a	PF20	——> DOWN
PF9	——> :b	PF21	——> SWAP
PF10	——> :i	PF22	——> LEFT
PF11	——> return;s.h	PF23	——> RIGHT
PF12	——> cursor	PF24	——> down max

CLIST edit macros are stored in source form in any PDS concatenated to SYSPROC. They are comprised of one or more of the following types of statements:

- Edit macro commands
- CLIST command procedure statements and comments
- ISPF and PDF dialog service requests
- TSO commands.

Although all statements are initially interpreted by the TSO command processor where symbolic variable substitution takes place, what happens next depends on which of the four types of statements it is. Each type is processed by a different component of the system; syntax rules and error handling procedures differ. Separate IBM manuals describe each type.

Each type of ISPF edit macro deserves a series of articles. There have been many articles published and large chunks of IBM manuals dedicated to the subject. However, there is a catch. Programmers like to program and it can become almost an obsession. Before writing an edit macro, ask yourself two questions.

- In the next year, will this save myself and any other staff member who might use it more than twice as much as it will cost me to code, test, debug and document it?
- Is this the best way to use my time right now?

Two edit macros were received from *MAINFRAME JOURNAL* readers. Tom Rusnak, systems programmer at C.P.S. Direct Marketing (Phoenix, AZ), coded an edit macro to solve the real-life problem posed in the article ''ISPF Techniques'' (February 1989). Instead of using COBOL sequence numbers to insert ascending numbers in consecutive lines of JCL, Rusnak inserted a member called PLUS in his CLIST PDS in SYSPROC (see Figure 2).

Typing //COPY1FIL EXEC TAPE1 FIL,FILE=! and then repeating it 120

FIGURE 2

```
ISREDIT MACRO (FROM,TO,BY,EXCL)
/*  The first occurrence of character FROM       */
/*  is replaced with the number TO, the          */
/*  second occurrence with TO + BY, etc.         */
/*  Normally omitted, specifying X or NX         */
/*  for EXCL will change only eXcluded or        */
/*  Not eXcluded lines.                          */
ISPEXEC CONTROL ERRORS RETURN
SET &RC = 0
DO WHILE (&RC = 0)
   ISREDIT CHANGE &FROM &TO &EXCL
   SET &RC = &LASTCC
   SET &TO = &TO + &BY
END
EXIT
```

times with the R119 line command, you would only have to type PLUS ! 1 1 to do exactly what it took 14 steps to do in the article.

In the VM environment, there are also two types of ISPF edit macros: EXECs and program macros. EXECs are written in either the REXX or EXEC2 command language and are stored in files with a filetype of ISREDIT. They are comprised of one or more of the following types of statements:

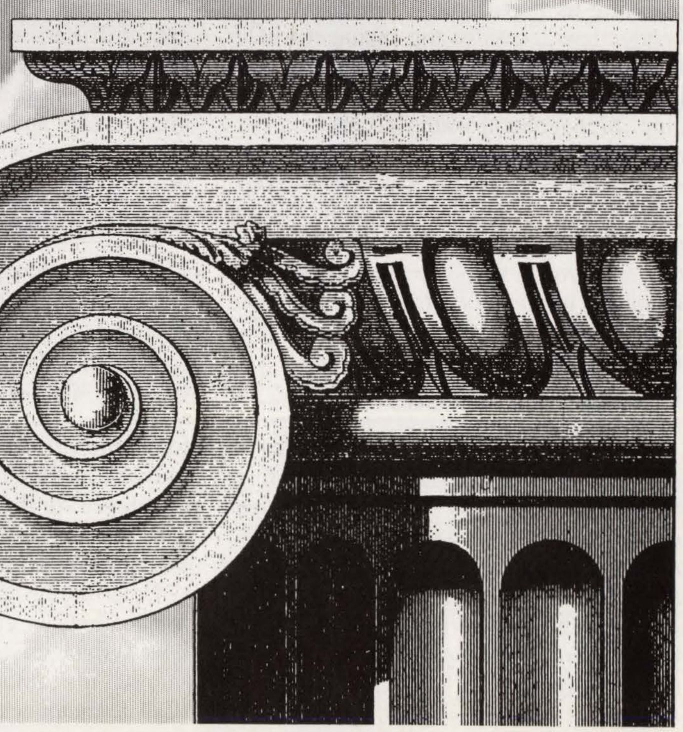
- Edit Macro commands
- REXX or EXEC2 command procedure statements
- ISPF and PDF dialog service requests
- VM/CMS commands and subcommands.

William S. Mosteller, CDP, Cross Product Advocate for Systems Center Inc. (Reston, VA), created an ISPF edit macro to send files he was editing to another system on his multi-node network (see Figure 3).

See ISPF *page 82*

FIGURE 3

```
/*  ISPF Edit Macro for transferring things     */
/*  to VMSI1                                     */
ADDRESS ISREDIT
'ISREDIT MACRO'
'ISREDIT BUILTIN SAVE'
ZEDLMSG = 'Return Code' RC 'from ISPF SAVE'
ADDRESS ISPEXEC
'ISPEXEC SET MSG(ISRZ001)'
ADDRESS ISREDIT
'ISREDIT (DSN) = DATASET'
ADDRESS CMS
'SENDFILE' DSN 'TO' USERID() 'AT VMSI1 (NOLOG'
```

8 Common Capacity Planning Mistakes And How To Avoid Them

By Mike Stackpoole

Most corporations enter capacity planning for the first time because of a change in business requirements such as sudden growth, a merger or a need for new technology. Whatever the reason, it is not uncommon at this stage for the vendor to do capacity planning as part of the sales effort. The vendor has access to top management in the corporation and the objectives of both the company and the vendor often coincide. Management wants to choose a system that will meet all the application requirements and the vendor wants to provide a system that will give his offering a competitive advantage. Often the vendor will work with the data processing manager to develop the plan and even participate in the presentation of the plan to top management.

Mistake Number One

It is not a mistake for companies to seek hardware proposals from vendors. *However, the error is in allowing the vendor to "own" the responsibility for planning capacity and in expecting the vendor to provide ongoing analysis of workloads.*

When the vendor makes the presentation to senior management, he takes responsibility for this particular hardware alternative. Rarely, if ever, will you see the vendor recommend a competitor's product over his own. The objectivity of the vendor has to be suspect at best and, at the least, he is strongly biased.

Capacity planning pays dividends six to 12 months before the critical need for new equipment. This foresight requires involvement with the development staff and end users to learn what new applications are planned in the future.

The vendor cannot provide this level of involvement and often makes his pitch for new hardware when you are in the midst of a capacity crisis. This is too late.

Mistake Number Two

The two major areas that will reveal the most relevant information to capacity planning are the business plans of the customer department and the monitor data from the application systems now running on the machine. The plans of the customer department will translate directly into the application systems that will be developed for them. It is these application systems that must become a key part of the capacity effort.

Determining the impact of these new applications is commonly referred to as *new system sizing* and focuses on determining the amount of system resource (CPU, memory and I/O) that will be needed to support the new applications. The temptation for the technician is to "fall in love" with the tools and do an overly detailed analysis of the current application systems. While they are an important part of the overall picture, they are *not* everything. Figure 1 shows a highly simplified diagram of the flow of information for the typical capacity planning effort.

Quantification of the potential impact of the new applications is a difficult task. It would be a mistake to involve the application customers (end users). The customers are not technically oriented, so you cannot ask them how many CPU seconds the new system will use. But it *is* fair game to ask how many people will use the new system (logon IDs)? How many concurrently active users do they expect (this cannot exceed the number of terminals)? How many terminals will be used? How much disk space will be needed? And finally, how many transactions do they expect to perform?

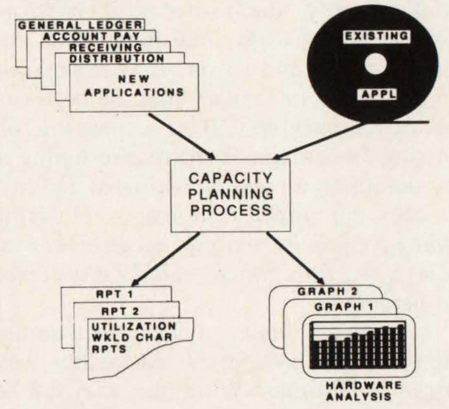
Each system will have its own definition of what makes up a transaction. The important item is to get an adequate translation from the business units to the computer units. If it takes three screens to add one new customer to the database and each screen is one complete transaction, then all you need is the estimate of the number of new customers and you are in business. You may get a blank look at first, but be persistent. Do not take *no* for an answer. Count the noses in the user department. The project leader or chief programmer can help with the disk estimates (record length, number of records). If you are using a relational database, see your database administrator.

Keep track of these estimates in a file as they will be invaluable later when you produce usage reports on Actual versus Forecast. Be careful here; these reports are intended to inform the person doing the estimating, not to ridicule them. The objective is to make them better forecasters over time.

Sizing the new application requires that you identify the resource consumption pattern of the *typical* transaction for the new system. You might consider breaking down the transactions into categories (light, medium, heavy). If you have no experience with the software that is being used for the new project, then you can look for other companies that have the needed experience. This is where your vendor can play a role. Either the hardware vendor or the software vendor should

FIGURE 1

Capacity Planning Process Model



be able to find another account that has already faced the same problem. Your network of capacity planning buddies is another source.

Once you find out how much CPU, memory and I/O is used by the typical transaction, multiply these numbers by the estimated volume of transactions (you get this from your customers) and by the concurrently active user count. This will give you a starting point. Finally, graph the estimates over time.

Mistake Number Three

The characteristics of a marketing person are not generally the same as those of a systems programmer or a capacity planner. But what are the characteristics of a capacity planner? There are six attributes as follows:

- Presentation skills
- Interpersonal skills
- Interviewing skills
- Architecture specific skills
- Statistical skills
- Financial planning skills.

It is difficult enough to find all of these skills in one person. But rarely, if ever, are they found in the technical staff. The technical staff can be expected to be proficient at interfacing with the computer system, as they should be. They usually fare quite well at the statistical and architecture specific skills. But customer interface, interviewing users, delivering presentations to top management and financial planning are not their forté.

In addition to the skill set issue, the technical services department is a busy place. *Allowing your technical staff to do capacity planning on a part-time basis is another mistake*. A part-time effort will not yield the needed results. It is almost impossible to anticipate the requirements of new systems on a part-time basis. Monitoring the general trends of current systems is not the same as watching CPU activity during the end-of-the-month rush. You cannot spot a change in the business trend from the screen of a performance measurement tool.

All too often, the part-time capacity planning effort is precipitated by a capacity crisis. This leads to reactive plans that simply announce to management that there is a need to upgrade the system. Technical personnel generally see their fate tied to knowing the latest technology. This attitude makes it natural for them to ask for an upgrade to the latest or next level of technology.

Mistake Number Four

Capacity planning is akin to baring your soul, in that *all* your assumptions *must* be explicitly stated in the presentation to management. *It would be a mistake to hide the real assumptions from management.* The estimates of business volume should, if you are doing it right, greatly impact the projections for new equipment. If these estimates change, you should also change the timing for the new hardware.

> *Capacity planning makes the assumption that the system is reasonably well tuned at all times.*

Growth rates are another area where there are often arguments over the validity of the assumptions. How can the computer department grow at 50 percent per year when the entire company is growing at 20 percent? You will need to anticipate this type of question and be prepared with an adequate answer. If you cannot defend the assumption, then you should change it. Perhaps 50 percent is not realistic. What are your most pessimistic and most optimistic assumptions? What are your assumptions for the economy? Is it realistic to forecast a no-growth scenario in the program development area when it is hiring programmers?

There are always assumptions made about the business, whether they are stated or not. If you do not state them and a senior executive asks about them — challenges them — in your presentation, you are doomed. You have just lost your credibility with top management whose position is that you are not in touch with the business and your recommendation is, therefore, suspect. You are far better off to explicitly state your business assumptions, have them reviewed by your boss or another trusted superior and correct any weaknesses prior to a presentation to top management.

Technical assumptions are the other side of the same coin. What is going to happen to the technology you are recommending? What are the vendor's plans for model changes? Will the investment in equipment be protected? Or is this the last of this line prior to a new and incompatible line? Are the transaction workloads based on the peak period for processing? The peak one hour average is commonly used in capacity studies as being an indicator of current utilization. How sure are you that the sample data used in the analysis is really representative of the true picture? The busiest time of day may well be from 10:35 a.m. to 11:35 a.m. But what is the trend over the month? Is the last week of the month four times busier than any other week? What are the quarterly variations? What effect do seasonal changes have on the projections? What is the difference between the daytime peak and the nighttime peak?

The biggest technical assumption is the one you make relative to system tuning. Can the system be tuned to last three to six more months? Is the system well tuned now?

The assumption for capacity planning is that the system is reasonably well tuned.

Mistake Number Five

What constitutes a financial analysis? How far into the future do you need to look? What methods need to be used? How does the proposed acquisition relate to the company budget? How much detail do you show to each level of management? Should the financial analysis be done by the capacity planner or the finance department?

Senior management wants to see plans that are consistent with the financial and strategic business plans and executable by the Information Systems (IS) department. So, if you are to have any chance of selling your recommendations to top management, you will need to correctly answer all of the above questions. *Skipping or glossing over the financial analysis would be a mistake.*

More than anything, a financial analysis should be complete — all costs must be included. The basis of the financial analysis is the tradeoff in costs versus benefits for each alternative solution. This basis must be over equal time periods for each alternative. If any one alternative uses three-year numbers, then all the alternatives must use the same three-year horizon. The analysis should include the difference in cost between new and used equipment and the difference in cost between purchased or leased methods of finance. Naturally, the time value of money must be taken into account.

How far into the future you need to look will depend on the particular configuration you are considering. Generally speaking, you will want to cover at least two upgrades. If you are considering a CPU upgrade, then include this upgrade and the next one in the financial analysis. The reason for this is that you can be deceived by vendors who only tell you the cost of a new machine. It may be cheaper in the long run to buy a smaller machine first and upgrade it to a larger processor in the future. This is especially true if the upgrade is available on the used market.

As alluded to earlier, the time value of money is an important feature of any financial analysis. A lease-versus-buy analysis that uses Net Present Value (NPV) techniques is a start. Be sure to include the cost of capital for your particular company and your hurdle rate. If you use a lease-versus-buy model from a leasing company, be sure to examine the calculation carefully to make sure it is not biased in favor of leasing.

Understanding the plans of your customers is a critical success factor in a capacity planning program. The budget cycle affords the capacity planner the perfect opportunity to be of value to the customer and to get to know his plans. The capacity proposals are most important during that favorite summer time activity, *plan time*. You should be able to provide to each customer a spreadsheet-style report showing hardware items on the left side and months across the top. Fill in the matrix with the cost of each item in the month it is planned to occur.

The amount of financial/technical detail shown to each layer of management is inversely proportional to each level in the organization. Less detail is presented to increasingly higher levels of management. Senior managers deal in millions with never more than one decimal point (4.3 million, not $4,321,298.44).

As to who should do the financial analysis, it depends on your organization. Some data center managers are terrified to let a *financial type* make a recommendation about their equipment. If the finance or purchasing department does not have experience in acquiring computer equipment and the IS staff does, then the IS staff should probably do the financial end of the analysis. The opposite is also true. If both have the experience, then a team approach may be used. If neither has the needed experience, then an outside consultant can be of great value. The important point to remember is that the presentation should contain the financial analysis; a technical analysis alone is not sufficient.

Mistake Number Six

Never provide alternatives.

The objective of the capacity planning function is to provide sufficient resources to meet service level objectives in a cost efficient manner. Many of you have read this definition and concluded that it is your job to determine what is needed and to tell management when it is time to upgrade the system. *That would be a big mistake*. The function of capacity planning is to provide options to management so that management, not capacity planning, can make the decision. Capacity planning is staff function, not line. As such, its job is to make recommendations, provide alternatives, do analysis, but *not* to make the decisions. If you want to make the decisions, get into a line position.

The capacity planner should provide at least three reasonable alternatives to management, including full financial analysis. Each alternative should include pros, cons, timing, cost and a short description. This is a good place for the one-page rule. All three alternatives should fit on one side of an eight-and-a-half by eleven inch piece of paper. Follow this with a recommendation and state your business reasons for making the recommendation. Make them relevant to your audience.

Nothing will turn off a manager faster than having a staff support person telling him/her what to do. They want options and it is your job to give them options. So do it and do it well.

Mistake Number Seven

Capacity planning is a long-term function that helps create hardware plans for the future. Performance tuning can positively impact those plans by improving system performance to the point where an anticipated upgrade can be delayed. Capacity planning on an ongoing basis is looking at a one-to-two-year horizon, forecasting growth, meshing with the budget cycle and so on. It is impossible to determine in January that the performance of a machine will be suspect the following December. Performance tuning is by definition a *right now* kind of activity. It does not forecast. It reacts. *Delaying doing a capacity analysis because the system is not fully tuned would be an error in judgment.*

Capacity planning makes the assumption that the system is reasonably well tuned at all times. While this may not be true, it is reasonable. And most important, it allows the capacity analysis to move forward.

Mistake Number Eight

The ability to articulate your thoughts, ideas and opinions in an intelligent manner is paramount to the success of the capacity program. *You do not want to give a poor presentation of the capacity analysis to management.*

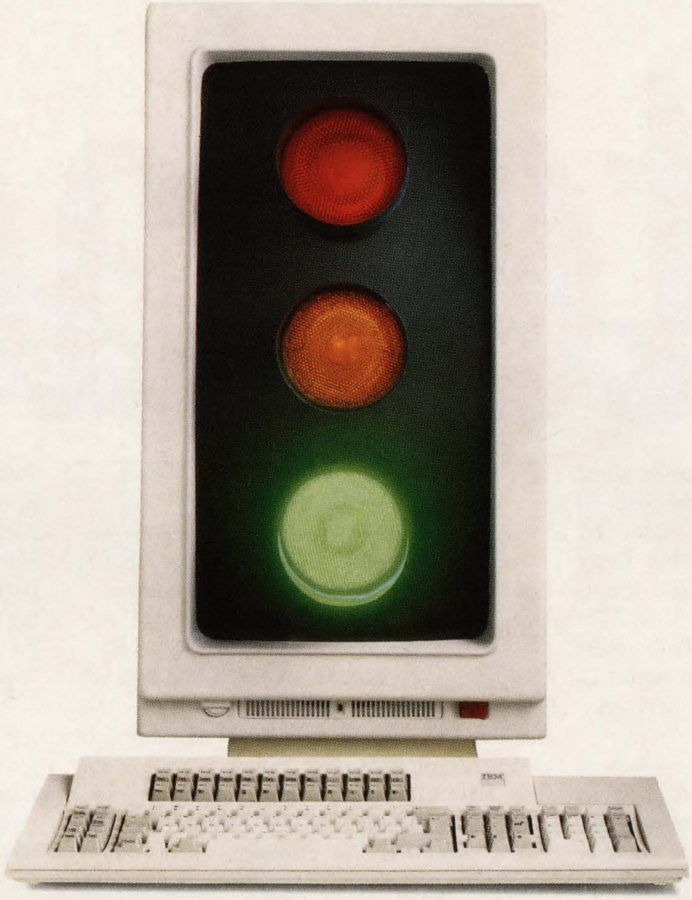
Management does not automatically know that you have worked 15 hours a day for the past month. It does not know you did data reduction, workload characterization, analytical queuing models, talked to a thousand customers and programmers, wrote until your fingers were worn off and so on.

What management does know is what it sees and hears at the presentation. The presenter should be relaxed in front of an audience. Use professional looking foils or slides. Make your points simple and direct. Do not overdo the gadgets. Keep it clean, simple and honest. Make sure the right people will be there. Try not to mix technical and managerial audiences. Verify the attendance of key management personnel a few days prior to the presentation. Give them comfortable facilities. Coffee is nice, but optional. Dress up for the occasion and practice your presentation several times.

ABOUT THE AUTHOR

Mike Stackpoole is president of Electronic Data Management, Inc. He has been in data processing for the last 18 years, focusing on performance tuning and capacity planning the last 12. EDM, Inc., 42638 Lerwick St., Fremont, CA 94539, (415) 438-9651.

All systems go.



"VISIT OUR BOOTH #86 AT CMG IN RENO — DEC. 11-15"

Meeting your service levels can be as hectic as making your way through rush hour traffic. Especially if you have multiple systems or multiple software environments.

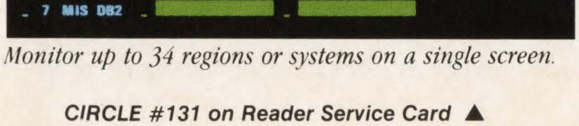
But with the Status Monitor™ from Candle Corporation, you can keep all the lights green for all your systems all the time. A single screen gives you a complete overview of your entire DP enterprise - MVS, CICS, IMS, and DB2. That's why more than 4,000 Candle customers are already using it.

The Status Monitor uses colored bars to tell you instantly which of your systems is in trouble (red), threatened (yellow), or doing fine (green). And when there is a problem, a single keystroke speeds you directly to the appropriate OMEGAMON®. For example, you can zoom straight into OMEGAMON for MVS to analyze the impact of performance groups on each other. Or to OMEGAMON for CICS or IMS or DB2 to find out why response time is slow.

Because the source of a problem in one system is often caused by applications in another system, the Status Monitor gives you complete freedom of movement across environments. If the cause of gridlock in CICS is not CICS at all, but DB2, you can move straight from OMEGAMON for CICS to OMEGAMON for DB2. Or back to MVS.

The Status Monitor is a component of OMEGACENTER™, Candle's system solution for total service level management. OMEGACENTER combines the precision of Candle's performance management software with cross-system, cross-environment monitoring. With the automation component, solutions are implemented automatically at machine speed, while the remote control component gives you complete troubleshooting capabilities wherever appropriate personnel happen to be.

To find out how to keep the lights green in your data center, call Terry Forbes today at **(800) 843-3970**.



Monitor up to 34 regions or systems on a single screen.

!Candle®

CIRCLE #131 on Reader Service Card ▲

VSE/VTAM Interfaces

By Leo Langevin

This article will give VSE users some understanding of the purpose of VTAM, the terminology used by the VTAM community and some basic knowledge showing how all the definitions fit together, as well as the use of network maintenance tools. This overview will not go into any great detail as to how to define or control the network.

What Is VTAM?

VTAM is an access method, just as VSAM is an access method. The purpose of an access method is to provide the user with the ability to perform a series of operations in such a generic manner that the application program need not know about the actual physical characteristics of the request. For instance, when you issue an OPEN, you do not really care about the order in which a series of programs are called to perform that operation. All you really care about is that you issued OPEN and the return code for that operation indicated that it worked.

Access methods are buffers existing between the application code and the Physical Input/Output Control System (PIOCS). For VTAM, you do not want to know the bit settings required to send an acknowledgement. In fact, you probably do not even want to have to provide the acknowledgment either. VTAM provides a series of macros that will perform a series of function calls that will perform a series of operations and inform the application of the result.

VTAM is a subsystem. That is, it has its own environment but is still dependent on VSE for its internal support and operation. VTAM can run in a POWER-controlled partition, but normally it is run in a non-POWER address space so that it has the highest priority in the system.

The VTAM Configuration

The configuration of a VTAM system is relatively easy. VTAM uses the SOURCE library of VSE to maintain all of its definitions. All source members are .B type entries. For instance, the start-up list member is by default called ATCSTR00.B. It contains information concerning the number of buffers used for various operations, tracing facilities and so forth. It also contains a pointer to another member, one that contains a list of all of the other nodes to activate.

A node is a definition in the network that is controlled by VTAM. A major node is normally the name of the *book* cataloged into the source library. It could be a definition for a series of terminals, application names or communication lines. For example, if I define a series of local terminals, I may catalog them in a single member LTERMS.B. When I enter the command D NET,MAJNODES to request a display of the names of all of the major nodes in the network, it would show LTERMS as an entry. Notice that the .B is not carried over in the name. It is only used in the definition and cataloging of the member.

As I said before, ATCSTR00 will have a pointer to a list of entries to define to the network. The name of this member is ATCCONxx.B, where xx is the two-digit suffix indicated in the ATCSTR00 entry. ATCCONxx will contain a listing of all other .B entries that should be automatically activated during system initialization, normally during an IPL. By executing the phase ISTINCVT, VTAM activates, looks for the startup options list, partitions its memory and then activates the desired major nodes. Figure 1 is a pictograph of that process.

APPLS And Other Delights

One term used in VTAM is an APPL, which is the name of an Access Control Block (ACB) controlled by VTAM. All programs that need to use access methods have an ACB defined. This is done via the ACB macro. It is by issuing an OPEN against the ACB that a request is made to VTAM to establish a link between it and the requesting application. After a successful connection occurs, the applications program may then issue such commands to permit terminals to logon to it, to send and receive data and so on.

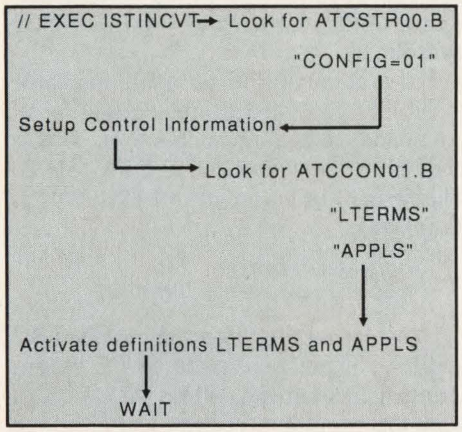
You can display all of the APPL IDentifiers (APPLIDs) by entering the command D NET,APPLS at the console. You define the APPLS to VTAM by defining a MAJOR node that contains an entry with a matching ACBNAME such as the one in the desired application. If you define a program called SHOWME, you need something like this:

```
CATALOG    APPLS.B   REP=Y  EOD=/+
APPLS      VBUILD    TYPE=APPL
SHOWME     APPL
RJE        APPL
           END
/+
```

This defines a major node of APPLS with the minor nodes SHOWME and RJE. SHOWME is now defined to the system. To activate the node after creating a new major node, use the command:

```
V NET,ACT,ID=APPLS,SCOPE=ALL
```

FIGURE 1



The SCOPE option tells you to activate the minor nodes.

Now that you have an active APPL, you can execute your program that uses the ACB of the same name. One thing to remember in VTAM is that all names must be unique. This means that the ACB name of one program cannot be the same as another and be able to run concurrently. Establishing proper naming conventions while the network is still young is one of the best things you can do.

If you make a change to an existing major node, such as adding or deleting a minor node, and recatalog the source member, the changes are not yet in effect. This is because when you issue an ACTivate command, each entry is formatted and stored in memory. In order to place a fresh copy in current status, you must release the current definitions and then reinstall the formatted entry with the following commands:

```
V NET,INACT,ID=APPLS
V NET,ACT,ID=APPLS,SCOPE=ALL
```

Be certain to wait until the first is complete before performing the second. Also, if you disable an APPL while an application is using it, such as CICS, you will terminate all VTAM sessions and possibly abend the application. Therefore, it is important to make sure that no sessions are active before performing the termination. By entering the command D NET,APPLS you will see the status of all applications. If the session is in NEVAC state, then the APPL was never activated and no harm is done. If CONCT is seen, then the APPL was activated, but there is currently no program issuing an OPEN against an ACB of the same name. INACT means that the APPLID was once active but has since been deactivated via the V NET,INACT command at an earlier time. And, of course, ACT means that the session is active.

After you OPEN an ACB and perform the various commands, you will need to CLOSE the APPLID before issuing an EOJ macro. If you do not, then VTAM notices that the APPL is no longer in a session and will issue an automatic CLOSE for you. What occurs is that two to three lines of messages appear, looking like a form of error and causing the neophyte operator to panic. Just for neatness' sake, all OPENs should be followed by a CLOSE.

The Logical "Don't" Get Physical

A couple of other terms that you should be acquainted with are Logical Units (LUs) and Physical Units (PUs). All PUs are those having a direct physical connection or relationship — normally communication lines, terminal controllers, communication controllers and so on. An LU is any node defined with an LU parameter as well as APPLIDs. Connection LUs are normally such things as session definitions instead of the actual physical device characteristics. An example of how LUs and PUs relate could be in defining a physical communication controller that can contain two logical sessions:

```
VBUILD
  PU
    LU
    LU
END
```

Access methods are buffers existing between the application code and the PIOCS.

You can see a list of all of the terminals defined on the system by entering the command D NET TERMS; however, this may be a bit lengthy and is not suggested for larger installations.

Your "Guest" Is As Good As Mine

A CPU is normally known as a host, because it contains not only an operating system, but also VTAM as part of the environment. If you are attached to the host, you are a *local* device. If you require a telephone line to enter the CPU, you are a *remote* device. If you are not the *host*, you are a *guest* of the host. So, a PC dialing in from home to access CICS would be a remote LU or guest. Remember, it is an LU since the host does not know of any direct connect to the PC; therefore, any session to an unknown is logical only.

Like any host, VTAM has a home or *domain* where it exists. When one host communicates to another host via a communication link, you have what is called *cross-domain* communication, since each is crossing the borders of its own home. This type of session is also called LU-to-LU in that these are not physical connections but logical links. More correctly, it is called SSCP-to-SSCP since each host contains a System Services Control Program that is controlling the required session for each particular side. Therefore, the PC-to-mainframe would be called an LU-to-SSCP session. And two application programs running under the same host but in communication with one another would be called LU-to-LU sessions, since an APPL is an LU and no SSCP is required.

For a terminal connected to one host to logon to an application at another host, there has to be connectivity by defining the Cross-Domain Resource Manager (CDRM), which provides a pointer down the PATH that ends up at the desired location. But you want not only to get to the *host*, but also to logon to the application or resource of that host. Therefore, you need to also define the Cross-Domain ReSourCe (CDRSC), so that when the terminal requests to logon to APPLID=CICS11, VTAM will look up the LUNAME and find that it is located in a CDRSC, which points to a CDRM, which points to a *path* through which it will follow a *route* until the terminal and application session are *bound* to one another, allowing the LU-to-SSCP session to be established.

VTAM is an access method that handles all of the PIOCS relating to transmitting and receiving data, as well as setting up and establishing session parameters, so that an application need not be concerned with the specifics of the network but, instead, can get on with the acquisition and generation of data. But once you have a network, how do you

FIGURE 2

```
NCCFAPPL   VBUILD   TYPE=APPL
NCCF       APPL     AUTH=(NVPACE,ACQ,PASS),PRTCT=NCCF
NCCFPPT    APPL     AUTH=(NVPACE,PPO), PRTCT=NCCF
NCCF000    APPL     AUTH=(NVPACE,SPO,ACQ),PRTCT=NCCF
NCCF001    APPL     AUTH=(NVPACE,SPO,ACQ),PRTCT=NCCF
NCCF002    APPL     AUTH=(NVPACE,SPO,ACQ),PRTCT=NCCF
```

keep it all together?

Networking Tools: NCCF/ NetView

Nowadays, you do not hear too much about NCCF. You can see references to it on some manuals and you can even find it listed in some of the ordering guides (as of 1989). But if you try to order it, you will find that it is no longer supported. What happened to it? Where did it go? Well, the answer is that it did not go anywhere; it was repackaged under the name NetView and now combines the facilities of what were several individual components into a single network-controlling entity. It makes sense, if you think about it. Why bother selling individual network tools for low prices, when you can sell the entire bundle as a single unit for a much higher price? It makes marketing sense and it is irritating. However, that is the way things are.

NetView uses the same phases, the same macros and the same library structures and is defined the same way as NCCF. So from now on, when I speak of NCCF, I speak of the main VTAM message subsystem and command processor. Such other components as NPDA, NLDM and so on are not going to be reviewed in this commentary, since they are more appropriately left to books that can provide greater detail by focusing on telecommunications rather than the operating system.

NCCF is a program that can run in its own partition, just as CICS does. And like CICS, it is a VTAM application. For NCCF to communicate with VTAM, it needs to open an ACB, just like any other program, which means that an APPLID needs to be defined for it. NCCF has a primary session; it is constantly running, processing time-delayed commands, receiving unsolicited VTAM messages and so forth. It also has a series of secondary sessions. These sessions are when a user requests to LOGON to the NCCF APPLID. NCCF will look for a secondary APPLID that is not in session and when it finds one, it will allow the user to LOGON with USERID and password. To illustrate, Figure 2 presents a VTAM APPL major node definition for NCCF, allowing up to three users to logon concurrently.

As you can see, when NCCF activates, it will use the APPLID NCCF to allow users to establish a session. It will hold them there, looking for a free APPLID of the same name with an appended three-byte number starting from 000 and incrementing until a match is found. Once found, that APPLID is opened and a subsession is established. The PPT APPLID is for the Primary Program Task. This is the session that will receive the unsolicited RUs, as designated by PPO in Figure 2. There can only be one PPO defined for any system. This means that if you run NCCF, no other application can be running in the same network control mode.

Once you logon to NCCF, you are able to perform a variety of functions, as described in the appropriate operator's guide. The commands available are limited by the security and by the imagination of the person responsible for the product. You are able to define a CLIST to NCCF. This is a member cataloged exactly like a VTAM major node, containing the special NCCF script language to perform a specialized function. For users who have written ICCF PROCS, the form of this command language should not be too difficult. For example, if you wanted to be able to enter TERM LTRM020 and have it convert the command to D NET,ID=LTRM020,E, you could have a CLIST that looked like this:

```
CLIST
&CONTROL ERR
D NET,ID=&1,E
&EXIT
```

By cataloging the above in a library that is searched by NCCF as TERM.B, entering TERM LTRM020 would produce the desired result.

Commands can also be from actual Assembler-based programs written by the user. With such a program, you can inquire passed parameters, process them, inquire or perform a routine and then output a display in either TEXT mode (line-by-line mode without cursor fields) or FULL-SCREEN mode (a full screen with defined fields and cursor positioning using 3270 command strings). They range from the simple display of the terminal and user identifiers to being able to perform POWER commands from an NCCF console.

For a command to be made available to NCCF, you need to update the list of commands that are available to it. This is entered in a member called DSICMD.B. The format for each entry of DSICMD is as follows:

```
<COMmand> CMDMDL
  MOD=<phase>,RES=x,TYPE=x
```

Consider taking the program DSIABC and have it function under NCCF by entering the command WHO. You do not want this program to be resident all of the time, nor does it fall within any of the specially defined types of NCCF command processors. So the entry for it would be coded as:

```
WHO CMDMDL MOD=DSIABC
```

There is also a series of IBM-provided command processors for NCCF. These are what give NCCF its power. For instance, DSIPFK will provide PF-key support, DSIVTP will send commands to VTAM, DSICKP will clear the screen and DSIENP logs the user off. The NCCF administrator can add or remove commands, provide synonyms for the user, customize PF-keys and, in short, make NCCF into a full-process system to the point of emulating a CICS system. To provide an interface that gives POWER commands, checks tape drives, responds to console messages, reviews errors and inquires the accounting log or dataset database, you have now provided the operator with the only terminal necessary for him or her to function. By running under POWER and providing the ability to read from the library to submit data for execution, there is little else required of the general operator.

Like PNET, NCCF can communicate with its own kind. You can even have multiple sessions running concurrently from the same terminal to multiple NCCF domains. The command processors that you use from a remote site, however, can only output TEXT screens. Full-screen I/O is not supported except for local functions.

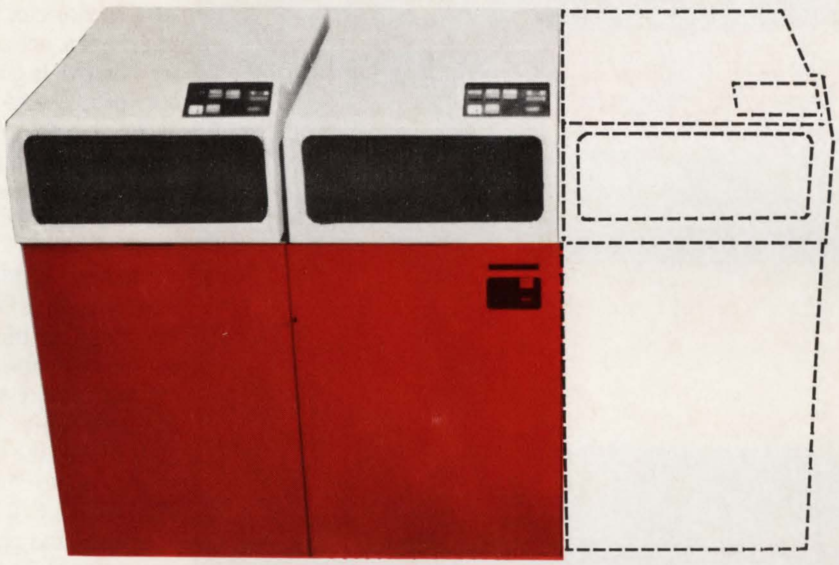
NCCF also has a built-in timer function. This means that you can set up certain commands to be processed at a specific time. Given the DSIPOWER command, you could use the command AT 16:00,PPT,PRELEASE RDR, BACKUPS to cause a job to automa-

PROBLEMS:

The DOS/VSE Label Area is a performance bottleneck.
Slow disk, relative to CPU, limits performance.

SOLUTION: BIM-VIO

The DOS/VSE 'Virtual' Disk Drive
and Resident Label Area Product.



Think of it as a disk drive with zero seek time and rotational delay, having no electrical, air conditioning, or floor space requirements!

BIM-VIO uses a facility in DOS/VSE/SP called "VIO" to map all I/O requests for a non-existent disk address to a virtual memory area outside the normal VSE address space areas. Since this area is in virtual storage, references to it are satisfied at CPU speeds and no actual disk I/O takes place, except possibly if memory paging results. The net result is a potentially significant performance improvement of programs using this area.

The virtual disk can be used for almost anything that does not require permanent retention beyond system start-up (IPL). Examples are compiler work files, SORT work files, temporary files used within or between application jobs. Application programs are not affected, the JCL is simply changed to point to the virtual disk drive 'address'.

A built-in aspect of the product is that the DOS/VSE Label Area is relocated to the virtual disk. This area is one of the most frequently accessed in most DOS sites, so moving it to the virtual disk should result in significant performance improvement to the overall system, regardless of any other specific use of the virtual disk capability.

BIM-VIO is priced at $4000 for a permanent license, $2000 on an annual lease or $200 on a monthly rental.

B I Moyle Associates, Inc. has been dedicated to providing cost effective software solutions, which improve system performance and user productivity, for over 10 years. For more information on BIM-VIO or any of our other quality software products and services, call Jim Kingsbury at 612-933-2885 today.

BIM

B I MOYLE ASSOCIATES, INC.
5788 Lincoln Drive
Minneapolis, MN 55436

612-933-2885
Telex 297 893 (BIM UR)

Member Independent Computer Consultants Assn.

CIRCLE #66 on Reader Service Card ▲

tically begin at a specific time. Or, you can cause a command to be processed periodically such as every hour using EVERY 1,PPT,command.

Here is one example of an operator application that I implemented under NCCF. I was in an environment where security was an important issue. The company wanted to control all job execution by having users use different IDs at a specific period. These IDs would be provided by the operations manager only. Under NCCF, you can limit which physical terminals can actually logon to it based on the VTAM logical unit name. Since the only people who could access NCCF were inside the computer room where no one was allowed and since this application to maintain the passwords was critical, it was decided to make this an NCCF-based application.

Since VSE does not use the FORMAT-5 label on the DASD (reserved for MVS usage only), this was decided to be the appropriate place for containing the encrypted/compressed information across the series of 3380s. To do so, I used the PUB table to find the CUU address, convert the VOLID and issue the VTOC macros OVTOC/PVTOC/CVTOC (see FILERENM) to update and read that area that was put into place. Once the batch version had been tested, the NCCF command processor version was written, providing full-screen support and the ability to cursor to a specific user for a single line change or the depression of a single PF-key (with verification) to generate new passwords, generate a report and segment the output to provide immediate printing at the local terminal printer.

As I said, the limitation of NCCF to support not only your network, but also your production environment is limited to your imagination. With NCCF you can automate functions, control your environment and make your operations staff very, very happy.

ABOUT THE AUTHOR

Leo Langevin is technical support manager for ITT, Bell and Gossett, Morton Grove, IL 60053. He has worked in data processing for 17 years, the last 10 in systems programming. This article is adapted from his book, DOS/VSE/SP Guide for Systems Programming, *published by QED Information Sciences, Inc., 170 Linden Street, Wellesley, MA 02181-0501, (800) 343-4848.*

Artificial Intelligence And Natural Expert Systems

By Roland Braun

Artificial Intelligence (AI) is still a new technology, not yet fully accepted as a valid field of science. The biggest problem in its general appreciation is the discrepancy between practical achievement and future promises. The overly optimistic pronouncements of early pioneers, who often seem to have anthropomorphized computers a bit too much, caused widespread disappointments and often emotionally loaded attacks on AI. But the error of these AI-pioneers is one of timing rather than of substance. They set out to accomplish within 50 years in machines what nature took millions of years to achieve in biological systems.

One thing has become clear from the reassessment during the last two decades: intelligence is more complex than initially envisioned and there are no magic shortcuts. The old truth that a computer can only do what it is programmed to do has asserted itself with a vengeance. However, after properly castigating the overly optimistic near-term projections of others, I am going to make my own, even more optimistic long-term prediction that the ultimate prospects of AI are overwhelming, going even beyond the most promising claims of past prognosticators. I also predict that this evolutionary process will be slow, much slower than earlier AI enthusiasts have forecast. Many decades will pass until true AI will emerge and many centuries before machines will come even close to the level of human intelligence.

To reconcile the seemingly contradictory assertions about present limits and future potentials, I will review current definitions of intelligence, knowledge and expertise. Also, I will take a closer look at biological expert systems which will reveal the following characteristics:

- Their design and behavior is controlled by a digital program
- The course of their evolution was slow and tortuous
- Deterministic systems preceded intelligent systems by a billion years
- The ultimate prospects of intelligent systems are spectacular.

The Nature Of Intelligence

Intelligence is an inherent ingredient in every human being; thereby, it is a common phenomenon. You use it routinely every day and think little of its complexity. So little, in fact, that some smart people started to claim that intelligence is just a matter of clever programming; that, with a little bit of luck and some magic hand waving, you could imbue computers with enough basic learning capacity and initial intelligence to have them acquire the rest on their own. So the scenario goes to the point that within a short time computers would become our equals in intelligence and soon surpass us by the sheer force of their speed, memory capacity, cloneability and immortality.

Casually overlooked in the quest for progress is the important fact that intelligence is not based on a single, static program. Intelligence is not a definable, unique mental capacity. It is the manifestation of many complex talents: the dynamic interplay of faculties like judgment, creativity, understanding and intuition. These depend on common-sense, experience and knowledge acquired over a lifetime of learning, which, in turn, depend ultimately on inborn instincts.

Also disregarded was the fact that intelligence depends on a specific biological structure, the central nervous system. As the most profound parallel-processing system known, the central nervous system encompasses a whole array of hierarchical control levels (from the conscious to the subconscious mind, from the voluntary to the involuntary nervous system) and it is intimately intertwined with a vast strata of innate instincts and biological drives that provide the subconscious with motivation and goals for consequent actions.

Even more significant, AI enthusiasts downplay the vital connection of intelligence and responsibility; intelligence without understanding, without motivation or moral apprehension, without a realistic model of the world and a moral conscience, would be of limited use, even perilous. The phantom of intelligent machines, turned loose without proper constraints, has caused some of the most vitriolic reactions from the academic community.

While the fear of computers taking over the world or of expert systems degrading human expertise and dignity appears unfounded, some words of caution seem justified nevertheless. As is generally the case, the more intelligent and complex a system, the less predictable its behavior will be and the more important the test and verification process becomes. As in any other context, the delegation of decision-making power incurs some risks. But with properly designed and updated systems, that risk should be no more than with sometimes inexperienced human decision-makers. Also, any large, complex system will not suddenly "take over" but grow slowly in an evolutionary process under close supervision and under continuous testing and adjusting in its working environment.

The Clever Hans Syndrome

Before the age of computers, man had a relatively simple definition of intelligence: it was the ability to deal with abstract concepts like language, numbers and mathematical relationships — the capability that separated man from animals. This simple definition was seriously contested when almost 100 years ago a horse appeared that could count and add numbers. Clever Hans demonstrated his gift all over Europe, tapping off numbers with his front foot. Some scientists were willing to concede intelligence to an animal if it could add two plus two!

But when it became clear that Clever Hans could not add at all, that it was by intuition that he decided when to stop the hoof-tapping, that his cue came from interpreting subtle facial expressions of his master or other bystanders, he was thrown back on the scrap heap of history in a hurry. Intelligence was reconfirmed as a uniquely human attribute and its accepted definition was saved for another day.

Things became a bit more tricky when computers arrived on the scene. There was no question any longer about the capability to count and add numbers. As a matter of fact, computers were so much better at handling abstract symbols that they were admiringly called "electronic brains" and expectations shot sky-high. The old "abstract concept" definition went out the window and a new one had to be found to keep man's uniqueness claim alive.

And, indeed, a conspicuous weakness was soon discovered: computers could not learn, had no judgment, no emotions nor could they recognize faces. In short, they did not possess what Clever Hans had plenty of: motivation, intuition and sharp eyes. However, with the horse long forgotten, a new definition of intelligence was proclaimed stating, in effect, that as long as computers cannot recognize faces, fall in love and act on their own, they will not be called intelligent. (Clever Hans would easily pass as intelligent under this definition and a cross between a horse and a computer would probably rate as super-intelligent!)

The syndrome aspect of the Clever Hans story lies in the fact that, as long as it is unchallenged, the valuation of intelligence is low. During his early history, man freely admitted intelligence to animals and even to inanimate objects. Only with the ascent of science was the line drawn between ourselves and everything else; only then did we claim sole ownership of intelligence. However, the bar was set so low that computers and counting horses could easily cross over to the human side. Only when so challenged was the definition of intelligence rethought and the bar raised enough for temporary relief. As a consequence, there have been so many redefinitions that cynics have concluded you will never see AI in actual applications. This is because whenever some AI feature moves from the lab to the application floor it loses its mystique and with that its claim to intelligence. After all, you know in your heart that anything mechanical and programmable cannot be intelligent.

Intelligence is more complex than initially envisioned and there are no magic shortcuts.

The Nature Of Computers And Artificial Intelligence

We all know that computers surpass humans in many areas that require intelligence. For example, nobody will deny that humans need intelligence to compute 46x78 and that computers are much better at such computations. Computers can also check your spelling, handle spreadsheets, implement cross-sectional tomography, compute payrolls and excel in data processing jobs, all of which require intelligence if done by humans. Since computers can do them better, these tasks are defined as "mechanical" or "programmable" and not as intelligent.

The definition "Programmable = not Intelligent" removes, once and for all, the temptation to admit intelligence to computers. Since, by definition, programmable means doable by computer and since computers are usually more efficient than humans, the above definition also implies that intelligence is detrimental to efficiency. Indeed, even humans are usually more efficient on a pre-programmed assembly line than when relying on their intelligence. Of course, everyone would be glad if the level of intelligence in computers could be raised to that of human assembly workers. However, the fact remains that even in humans with their superb intelligence, routine (pre-programmed) reponses are faster and more efficient than responses controlled by intelligence.

This basic inefficiency principle is further magnified by the fact that today's computers are bred for conventional, serial programs and that any intelligence given them is of low caliber. Indeed, computers are such formidable performers in mathematical and processing jobs that their much slower and more memory intensive operation in rule interpretation and logic reasoning makes AI applications appear even more inefficient by comparison. Only due to the enormous advances in cheap processing power and memory size has the practical application of AI and expert systems become realistic on a broader basis.

To illustrate the relative inefficiency of AI programs look at a simple example: $X+Y=R$. Any computer can derive R for any values of X and Y and the programming effort is negligible. However, if you would have to force this simple relationship into expert system rules, it would require infinitely many rules of the form "if $X=x$ and $Y=y$, then $R=r$" to cover all possible X and Y values. Nobody would put such a mathematical expression into production rules, of course. However, if $R=f(X,Y)$ did not follow any mathematical law, then you would have to use just such multiple rules; an expert system with an inference engine would be the best vehicle to handle such an irregular or heuristic relationship.

While computers are vastly superior in executing programmed instructions and while AI may make them more user-friendly and ever more capable and useful servants of man, computers are still totally inferior to humans in terms of true intelligence (for example, understanding, common sense, judgment, intuition and so on). To be sure, you can discern rudimentary intelligence and elementary reasoning in many AI programs, but there is no program in existence yet with responsible intelligence or with even a faint understanding of the data it can process so efficiently.

AI has developed in relative isolation from the complementary studies in psychology and neuroscience, though its ultimate goals and scientific roots are (or certainly should be) similar. According to some biologists, AI has proceeded on the flawed epistemological assumption that the brain can be understood in terms of a relatively simple universal computing mechanism. While AI is certainly more complex than requiring just a simple computing system, it does not need magic either. The only way to realize AI will be along a slow, evolutionary path that will require a lot of hard work, continuous testing, modification and verification, according to the simple formula:

$$AI = L\ (NI + HW + ES)$$

(L = Lots of, NI = Natural Intelligence, HW = Hard Work, ES = Evolutionary Selection).

Natural Intelligence And Biological Expert Systems

AI, at today's application level, is just a big word for the capability to extend data processing toward knowledge processing and reasoned decision making and to make computers more user friendly. Whatever intelligence and great performance there is, or will be, nowhere is there any magic or spiritual quality involved — not a trace of that mystery that is still so widely associated with human intelligence. Even in the most sophisticated expert system or AI program, everything is based on logic and natural reasoning principles; everything is man-made; therefore, by definition strictly mechanistic.

Do not stop there. Apply similar scrutiny to NI in order not to fall prey to the other extreme: the notion that AI is *not* intelligence at all and never will be because it is fundamentally different from NI. Realize that human intelligence also derives from a mechanistic program: it depends totally on the brain and through it on our genes, which are mechanistic and evolved from lower-level biological forms. Human intelligence is, therefore, not mystical either, not even unique, just more highly evolved than other forms of biological intelligence but still pre-programmed in our genes.

To get a better feel for the far-reaching implications of this reality and to gain a better insight into the quality and genetic origin of NI and into our biases and prejudices, look at NI in its most basic manifestation — an apple seed.

The original microscopic germ cell in an apple seed contains an expert system.

Nobody would call an apple seed intelligent or especially knowledgeable. However, it contains more knowledge about biochemistry and apple production than any human expert. A human would require a lot of knowledge and intelligence to produce apples. To produce them with the sun as the only energy source and soil and air as the only raw materials would be a feat worthy of a Nobel Prize. However, to build an automatic apple factory from scratch on a given spot with no resources other than what can be derived from the immediate neighborhood, would go far beyond human intelligence and human creativity.

Imagine an expert system that could construct such a factory, extracting the energy and all required materials from its environment, building first the extraction mechanism itself, then, piece by piece, the factory and production machinery and finally producing apples in quantity and all this without any assistance. Such a system would be the sensation of the century and everybody would agree that it requires intelligence and knowledge beyond human capacity, even beyond human comprehension. It would be equivalent to a miniature program disk with a built-in microprocessor that would start building its own computer and I/O devices in order to execute its specific application program and build a product.

The original microscopic germ cell in an apple seed contains just such an expert system: an expert system with a large database, an inference engine and a miniature protein production machine as its output unit. It also has a sophisticated input interface through which it receives information for deciding when to start growing and when to commence all its later activities. It has all the knowledge in its database for building a fully automatic and totally self-sufficient apple factory. It knows how to construct leaves that can produce carbohydrates using sunlight as its energy source, hydrogen from water and carbon from the air as its raw materials. It knows exactly how to use these carbohydrates to build a growing tree: a wooden trunk that can transport water up to the leaves and food down to the roots; a root system that holds the tree in place and draws water and minerals from the soil; and more leaves to make more building blocks for all this construction work. The expert system also knows how to recognize and fight parasites and how to create those exquisite apple blossoms with the right fragrance and color to attract bees for pollination. It knows when to produce blossoms for the first time and every year thereafter, dependent on weather conditions. It also knows all about sexual reproduction and how to transform blossoms into apples of the precise type that contain new seeds to keep the apple business going forever.

Such knowledgeable performance is absolutely fantastic. And there is no question that it all derives from the microscopic DNA molecules inside the original seed in intimate cooperation with the natural laws of biochemistry and in precise anticipation of the range of environmental conditions and support systems like water, minerals, air, soil, sun and bees. In plants such knowledge is attributed to a genetic program, in animals to instincts and in humans to intelligence.

Another most revolutionary aspect of the expert system is its start as an almost pure software system that builds its own hardware. A copy of the original software (the DNA program) is included in every hardware cell so that each cell can implement its assigned function with great flexibility under changing conditions. A given cell can start growing a root, twig or blossom, depending on external conditions. A small twig can sprout roots and develop into a separate new tree. This functional parallelism is remindful of the massive parallelism of neutral networks and has similar far-reaching consequences.

You have known the staggering capabilities of biological systems for some time and have seen them at work since the dawn of our own existence. However, as long as such superhuman abilities are confined to seeds and fertilized eggs, as long as they happen routinely all around you, not much notice is taken. They were called ''miraculous'' as long as you did not understand them and ''mechanistic'' as your insight grew. No one thought to attribute knowledge or intelligence to seeds. After all, their actions are precise and predictable, based on a genetic blueprint in their DNA molecules and on the biochemical forces of nature. Their performance is, therefore, not intelligent but mechanistic, based on mechanistic principles and natural laws.

The Nature of Human Intelligence

Humans have many different kinds of intelligence. The higher forms are usually called intuitive and creative. At the other end lie the deductive and recollective forms (knowledge), characterized by mechanistic complexity (mathematical proficiency, expertise, instincts and so on). The inclination is to rate these as lower forms or exclude them entirely from the realm of intelligence. However, they cannot be separated easily because of the intimate relationship with their counterparts: there is clear evidence that these lower forms are direct precursors of the more esoteric forms and vital prerequisites for any kind of practical intelligence.

Even if you could separate them, the lower forms of intelligence would still require special valuation. This is because much of conventional wisdom, human competency and common sense, as well as technical and cultural achievements are built on them. For example, nobody could create a modern computer or a jet airplane by intelligence alone. Even the most intelligent person could not have built any of these modern machines a hundred years ago before the necessary knowledge and expertise were available.

However, the most important question relative to AI is whether there is any fundamental difference between the various forms of intelligence and whether any may somehow be beyond the possibility of programmability. The answer to both questions, based on the evidence of biological systems, seems to be a clear *no*. Since all forms of human intelligence are derived from the same genetic program in the DNA molecules and since the brain is clearly a mechanistic organ and the exclusive seat of all higher intelligence, it appears that intelligence, in all its forms, is mechanistic and programmable.

Being able to implement true, high-level intelligence in computers is far from being realized.

An exhaustive explanation and proof of the mechanistic nature of NI is beyond the scope of this paper. However, if you accept that the lowest life forms (at the level of the apple seed and below) are mechanistically defined, then there is little doubt that also the higher animals and their NI are so defined. This is because there is a seamless continuum from the lowest to the highest biological systems, including man. A good case can be made that all forms of NI are actually based on instincts — the distilled, billion-year experience of successful forebearers. Interests and motivations are largely determined by instincts; what, when and how you learn depends to a great extent on this innate program. You learn to smile, walk, talk, fall in love and reproduce according to an inborn time schedule and under instinctive guidance.

A four-year-old toddler has already learned most of the extremely complex intricacies of a natural language (the structure of which he is unable to understand), while the much simpler concepts of basic arithmetic escape him completely. A four-year-old child far exceeds the capabilities of the most advanced supercomputer and the most sophisticated AI program in the area of language understanding; while in the area of arithmetic his knowledge ranks below that of a simple calculator — all due to instincts that dictate learning priorities and learning capacities and, thereby, knowledge.

Even intuition and creativity, those pinnacles of human intelligence, are not as esoteric as it may seem. On closer inspection you find that your intuition is clearly based on instincts and personal experience; while creativity rests, in turn, on intuition, knowledge and reasoned recollection and on instinct-driven imagination and playfulness.

As a matter of fact, intuition and creativity seem to be more evolutionary stop gaps than ultimate culminations of human intelligence. After all, you need intuition most in situations of uncertainty. When you have all the facts and time for a reasoned response, you no longer rely on intuition. You can say that birds build their nests by intuition, while humans build their houses by knowledge. Similarly, if you had perfect knowledge, creativity would no longer have much value. If you know the optimal solution to a problem, no amount of creativity can improve on it. That it is often hard to define an optimum or impossible to attain perfection does not detract from the fact that creativity is of a lower order than perfect knowledge.

So it appears that our high-level faculties are not original creations at all. Rather, they are derivatives of innate instincts that you normally consider the lowest of your mental assets and the only ones that you freely admit as being tied to a genetic program. In reality, your instincts may turn out to be your most valuable possession, representing the collective memory of all the positive experiences that helped your ancestors to survive and evolve. Instincts are essential for living (infants totally depend on them) and for social conduct (feelings of respect, love, compassion and moral conscience are based on them). In-

See AI page 82

SQL/DS Database Reorganizations

Technical And Operational Considerations

By Michael Teitelbaum

There are several activities connected with maintaining a healthy, reliable and efficient SQL/DS database environment. These include the following:

- Data reorganizations to provide the best possible performance for query and insert activity
- Database archives to provide point-in-time protection from database corruptions, DASD failures and otherwise non-recoverable user errors
- Table/DBSPACE backups to provide snapshots of data at various points in time; these can be used to recover from application/user errors where point-in-time recovery is either not a requirement or is not feasible.

With read/write applications, it is imperative for good performance that tables/ DBSPACES be analyzed on a regular basis to determine if they should be reorganized. *Failure to do so can lead to severe deterioration in performance*. A user may complain that a transaction that used to be quick (or at least was not slow) is now slow. This can often be an indication of a table/DBSPACE needing reorganization.

The steps necessary to reorganize SQL/ DS data are well documented; however, what is not as well documented are several considerations. One is a clear explanation of why reorgs are important and, therefore, an understanding of the ramifications of not doing them.

Criteria for deciding what, as well as how often to reorg is another one.

Third is the operational issues connected with implementing a system to do reorgs such as system availability, cost/ chargeback issues and DASD/tape availability to support the temporary files created during the reorg. Last is descriptions of additional steps that can be performed to improve the efficiency of the reorg process.

The purpose of this article is to describe some of the operational and technical issues associated with implementing a database reorganization system. It will be done in the context of describing and comparing two methods of accomplishing reorgs, highlighting those areas where technical choices may be driven by operational requirements. Possible scenarios for implementing a reorg system will also be described.

Reorganization

Reorganization has several meanings. It means reclustering data in the order of the first index created. A well-chosen clustering order can provide significant performance benefits for many types of queries.

Reallocating free space on data pages to support quick insert processing and the efficient processing of VARCHAR expansions and population of NULL columns is another meaning. Last, it means dropping and recreating all indexes to remove fragmentation and regain dead space.

The need to periodically reorganize data is not specific to SQL/DS; it is a necessary database maintenance task for all DBMSes. The reorg process itself is fairly straightforward. To review and compare two possible methods for doing a reorg, assume the following. *The reorgs run while SQL/DS is in Multiple User Mode (MUM),* allowing tables not being reorged to still be accessed. *The reorgs will be done at a DBSPACE level*. If there is more than one table in a DBSPACE and one needs to be reorged, rows from different tables are probably mixed on the same page and this may impact performance related to any table in that DBSPACE. The last assumption is that *reorgs will be done only in recoverable storage pools*.

STEP One — Determine The DBSPACES To Be Reorganized

This is really a two-part process. The first part involves running UPDATE STATISTICS. (UPDATE ALL STATISTICS is not recommended unless it is an ad hoc environment with many compound key indexes or there are many queries with predicates against non-indexed columns.)

This will update the statistics contained in the system catalogs for the tables, indexes and DBSPACES in the database. *It is required information for determining reorg candidates*. Run it against all the DBSPACES in the system that contain user data as well as any support DBSPACES (for example, QMF). UPDATE STATISTICS against the system catalog DBSPACE is also a good idea to help queries against the system tables. Read-only DBSPACES can be skipped.

The second part is choosing the DBSPACES to be reorged. This is done by analyzing the system catalog tables.

Criteria For Determining DBSPACES

Number One

The CLUSTER column in SYSTEM.SYSINDEXES for the *first* index created on any table in the DBSPACE is a "W" instead of an "F." A "W" indicates that SQL/DS does not consider the data to be clustered (physically grouped) in an order that closely matches the key sequence defined for the index.

The first index created is important since SQL/DS considers it to be the clustering index. SQL/DS always tries to maintain physical clustering of data according to the key sequence of the first index. When a new row is inserted, SQL/DS attempts to put the new row on the same data page as the row that is closest *to it according to the key sequence of the index.*

It is possible that the physical order of the data may match the key sequence of more than one of the indexes defined on the table. In this case there is still only one clustering index (the first one), but there is more than one clustered index.

Whether data is *considered* to be clustered or not clustered by SQL/DS has a major impact on decisions the Optimizer subcomponent makes when determining the best access path for processing queries that require multiple rows to be accessed sequentially. Along with the amount of free space available in the DBSPACE, it influences how quickly inserts of new rows are processed. Why is this?

When a *group* of rows will be returned as a result of a query and the query contains predicates that match a clustered index, the SQL/DS Optimizer is likely to choose that index as the access path to the data. Since the rows are grouped together on the same physical page, the number of rows returned per I/O will be high. *The fewer I/Os that need to be done, the faster the answer set is returned to the user and this ultimately is what good performance is all about.*

Under existing releases, SQL/DS considers data to be either clustered or not clustered — there is no middle ground. If the data is really not clustered, it is possible that as many as one I/O per row would be done to return the answer set. If there are only a few rows to be returned, there will probably not be any noticeable performance degradation. But, if there are many rows, it increases the possibility that the Optimizer may not choose that index as an access path especially when it is a non-unique index.

It may instead choose what it thinks is a better path such as a DBSPACE scan. This would most likely cause *unnecessary* I/Os to be done since it is probable that the data was not *totally* unclustered to begin with — but *the Optimizer has no way of knowing that.* If it did choose that index as the access path, additional *necessary* I/Os would still be done. So, either an incorrect assumption of unclustered data by the Optimizer or data that actually is unclustered can result in a noticeable deterioration in response time.

It is worth noting that under V2R2, the latest release of SQL/DS that has been on the market for more than a year, there is a new system catalog column called CLUSTERRATIO. It will contain a ratio that accurately describes the degree to which the table is clustered. The information stored in this new column effectively removes the all-or-nothing restriction. The Optimizer has been enhanced to take this new information into account, thus increasing the probability that it will continue to pick the clustered path, where appropriate, for longer periods of time.

Insert processing is impacted since, as was noted above, SQL/DS attempts to physically store a new row next to the row identified by its closest key value. If there is no room on the page, SQL/DS will begin searching for the closest page that has room. As more inserts (and updates) are processed, the searching happens more often and takes longer. Eventually, the data is considered to be unclustered.

Here is a summary of all the possible values for CLUSTER in SYSINDEXES: Remember, this CLUSTER value is as of the last UPDATE STATISTICS.

"F" says: I am the first index created and the data is physically stored in the key sequence defined by the index (clustering and clustered).

"W" says: I am the first index created and the data is not physically stored in the key sequence defined by the index (clustering but not clustered).

"C" says: I am *not* the first index and the data is physically stored in the key sequence defined by the index (not clustering but clustered).

"N" says: I am *not* the first index and the data is *not* physically stored in the key sequence defined by the index (not clustering and not clustered).

For these reasons, data that is out of cluster, as indicated by the CLUSTER column value for the first index, is probably the most important indicator for determining when a reorg should be done.

Number Two

The number of *overflow* rows indicated in SYSTEM.SYSCATALOG is greater than zero. In SYSCATALOG there is a column 'NOVERFLOW' that indicates how many rows in the table have moved (overflowed) from their original physical position to another. This occurs when VARCHAR or NULL fields grow in size and there is no room on the original page to hold the new row. In this situation, SQL/DS will move the entire row to a new page and place a pointer at the old row location noting that it has been moved. This is done to support an SQL/DS rule that once a row is created its row identifier, which is made up of DBSPACE page number and offset into that DBSPACE page, will not change. This prevents the excessive overhead that would occur processing changes to index pages as a result of row IDs being changed. Since the index only knows the original row ID, overflowed rows are to be avoided as it is possible that double the number of I/Os will be done, even if the index is clustered.

Number Three

There is evidence of *index fragmentation.* Index fragmentation results from SQL/DS splitting index pages in order to maintain physical clustering within an index as new values are added. Whereas data pages may become unclustered, index pages must remain physically clustered by key values. To maintain this index clustering, SQL/DS dynamically splits index pages in half when it needs to make room for a new key value. Fragmentation does not necessarily mean that there is no physical space, it means that there is insufficient space left where it is needed.

Fragmentation can occur over time with normal daily processing, but it is also seen in situations where bulk loading is done and there is an index that does not match the order of the data. If there is a large volume of data, fragmentation can easily take place. Alone with performance considerations, this is one of the reasons that indexes should be dropped before doing a bulk load.

The most accurate method for detecting fragmentation is to issue the SQL/DS operator command SHOW DBSPACE n where n is the number of the DBSPACE about which you are requesting information. This command is resource intensive as it does a DBSPACE scan. The information is informative, showing the number and percentage of index and data pages that are allocated and occupied. It will also show the average amount of free space on each page. Usually, in a fragmentation situation a high percentage of free, as well as occupied, pages will be seen.

While dropping and recreating the indexes will relieve fragmentation, it is wiser to do a reorg accomplishing the index recreation as well as reclustering and reallocation of free space. If you find that fragmentation is taking place, but the data is *not* out of cluster (CLUSTER = "F" or "C"), take a fresh look at the amount of free space that is allocated to the *index* pages (as opposed to the *data* pages).

In addition to select processing, fragmentation also negatively impacts insert processing. How much it does depends on the number of indexes for the table as well as the free space that is available on each index page. For each new row added to a table, all indexes must be updated. Each time an index must split a page, there is additional processing time taken to complete the insert. The more indexes there are, the greater likelihood of a problem.

A related index problem has to do with insufficient space available in the index due to deleted rows. When a row is deleted from a table, the corresponding index entries are also deleted. However, this space is not available for reuse. The only time this space is reclaimed is when the index gets dropped and recreated.

STEP Two — Running The REORG

This is accomplished through user written routines or SQL/DS Database Services Utility (DBSU). DBSU, using the UNLOAD and RELOAD facilities, is the method of choice since it can provide the function needed with a small group of commands. The steps involved are the following.

Unload

This causes a SELECT * ORDER BY the "F"/"W" index to be issued for each table in the DBSPACE. This forces the data to be sorted in the order of the first (clustered) index. There is a technique for avoiding this sort and thus reducing the time it takes to do an UNLOAD. Update the cluster column in SYSINDEXES for the clustering index, changing it from a "W" to an "F." This fools SQL/DS into thinking it is clustered and then choosing that index as the path for unloading the data. Usually, SQL/DS will do a DBSPACE scan to read every row in the DBSPACE and then, depending on buffer and table sizes, externalize these rows to DASD in order to perform a sort. This greatly increases the time to do an UNLOAD as well as requiring sufficient free pages in the storage pool that supports internal DBSPACES.

Reallocate Free Space In The DBSPACE

This is done by adjusting the PCTFREE value through the ALTER DBSPACE command *before and after the reorg*. At this time, increase the value above zero percent, forcing free space to be allocated on each data page during the RELOAD. This can be done at any point up until now in the REORG process, but it must definitely be done prior to the RELOAD (next) step. Deciding how much free space to allocate can be a complex process, but it can be simplified and an educated guess can be made with several pieces of information as follows.

- Average row length — the Effective Page Size (EPS) that is used to calculate the number of DBSPACE pages that will be needed has row length and free space allocation as part of its formula.
- Percentage of access that will be inserts and updates — along with the frequency that reorgs will be performed, this impacts free space settings.
- Distribution of inserts — if the inserts are only at the end of the table (as defined by the "first index"), then you do not need any free space. Free space for inserts is only necessary when rows will be inserted *between* existing rows.
- Expansions of rows due to VARCHAR or NULL fields being populated — to avoid overflows, space needs to be allocated to support these operations.
- Frequency of reorg job cycle — how often will this DBSPACE be checked to see if it needs to be reorged? If it is not often yet update, insert or delete activity must be supported, then you will want to allocate a larger amount of free space. This will allow for longer intervals between reorgs since there will be more room available on a given page for changes. The trade-off for less frequent reorgs is increased DASD usage and possible performance degradation when selecting or changing a *large* percentage of the table. This is due to the increased number of I/Os necessary to process the same amount of data now spread across more pages.

There must be sufficient physical (storage pool) space as well as sufficient logical (DBSPACE) space. If not, SQLCODES -701 (no data space) or -702 (no index space) will occur. In the *System Planning and Administration* manual, there is a step-by-step explanation for estimating DBSPACE data and index page sizes. This formula could be translated into an EXEC or a program and then DBSPACES could be checked prior to the reorg. Sufficient storage pool space could be verified by issuing a SHOW DBEXTENT command from the EXEC or program.

PCTFREE means the minimum, *not* exact amount of free space that will be allocated. Depending on the row length, it is possible to have *different* PCTFREE values and still have the *same* number of rows on each page. Free space management in SQL/DS can be confusing to understand at first, but it is important that it be understood. With some experimenting and a thorough review of how SQL/DS manages free space, reasonable numbers can be determined. The *Diagnosis Reference* and *Database Planning and Administration* manuals describe free space management.

Execute The RELOAD

RELOAD comes in two flavors — PURGE and NEW. PURGE is actually several steps executed with just one command.

For each table in the file, RELOAD PURGE processing drops the indexes and deletes all the rows from each table in that DBSPACE. It continues by inserting the sorted rows from the unloaded file back into the DBSPACE, one table at a time. The indexes are then recreated and processing finishes with an UPDATE STATISTICS being executed.

Step Two — Running The Reorg will be continued in the next issue.

ABOUT THE AUTHOR

Michael Teitelbaum is the president of Michael Teitelbaum Associates Inc., a New York City-based consulting and training company specializing in SQL/DS. He has worked with SQL/DS since 1984 and has been actively involved with the SQL/DS committees at SHARE. In addition to teaching and consulting he is also the chairperson of the New York City-Metropolitan Area SQL/DS User's Group. 175 W. 79th Street, NY, NY 10024-6450, (212) 799-2200.

AI *from page 78*

stincts appear to be the only spontaneous ingredients of the human mind — the fertile soil from which spring higher level mental faculties. Instincts are the core of humanity, the essence that separates mankind from the computers. So it will probably be some equivalent of your vital instincts that you have to bestow onto computers in order to raise them above the level of unthinking robots and prepare the foundation on which to build true AI.

Conclusion

In spite of all the advances in computer technology over the last 40 years, being able to implement true, high-level intelligence in computers is so far from reality that any such pretense may appear unjustified and frivolous. Indeed, names like ''artificial intelligence'' and ''expert systems'' are somewhat misleading: there is no spontaneous intelligence or inherent expertise in any computer. Whatever there is has been introduced by a programmer, deriving from and trying to mimic, human intelligence. Progress toward true AI will be slow, based on a lot of hard work and on a tedious evolutionary selection process. You will probably see more exotic concepts, like fuzzy logic, and develop symbiotic systems, combining conventional processors with massively parallel networks.

In recognition of the great advances made in deterministic programs and conventional computer technology, and in view of the slow progress in AI, it is of interest to realize that nature has spent about three billion years on the perfection of deterministic, instinct-driven systems before venturing into the development of intelligent beings (who are still enmeshed in a dense web of instincts as well). On a relative time scale of one day, nature used about 12 hours just to develop its deterministic DNA system, then went for 11 hours with DNA-based instincts alone. During the last hour it added a touch of intelligence for the early mammals and experimented for about one minute with some higher forms of intelligence. Only during the last *one* second out of the 24 hours of evolutionary history has human-level intelligence existed on earth.

The two most important ingredients for progress in AI are learning and understanding. These are traits shared with animals and they seem to be clearly related to human instincts. In fact, they are a prime example of how important those instincts are. Without a selective motivation to learn and understand, without some basic common sense and a moral frame to assess the validity and relative importance of individual parts within an infinite search space, those traits would be ineffective and risky. Nobody would seriously suggest, for example, that computers learn by themselves and change their programs and activities without extensive safeguards and a proven record of predictability; that is, without a program akin to human instincts.

The fear of machines taking over the world provides a vivid illustration of the crucial importance of instincts. It is based on the misconception that AI contains human instincts. In reality, man is a product of a different evolutionary path. To survive and evolve, man had to conquer and kill; to prosper, he had to dominate and rule. Over millions of years, the associated survival instincts have become part of the genetic program. You will never impart such ''negative'' biological drives or instincts to computers that have no need for them. Without those instincts, computers have no incentive to dominate anybody or to take over the world. They will only do what they are programmed to do and remain perfect servants as long as they are programmed properly.

According to John Sowa in the introduction to his book, *Conceptual Structures, Information Processing in Mind and Machines,* ''AI is the engineering counterpart (of) cognitive science (which) is a merger of philosophy, linguistics and psychology, with a strong influence from computer science.''

A major contribution that biological and cognitive science can make for the advancement of AI is to prove the hypothesis that all forms of intelligence arise from innate instincts which are firmly rooted in genes. Therefore, intelligence is based on mechanistic principles, is programmable and is within the range of artificial implementation across the entire spectrum. Providing the equivalent of an instinctive value reference system is a prerequisite for broad-based advances in AI. Showing that deterministic systems with only low-level intelligence can achieve extraordinary results, as exemplified by the apple seed, can further help to provide not only a road map, but also an indication of the ultimate potential of AI.

ABOUT THE AUTHOR

Roland Braun is a senior engineer at IBM's Development Laboratory in Endicott, NY.

ISPF *from page 65*

Mosteller explains that the macro saves the file currently being edited and then sends the saved file on to the system that will use it. Note that the SENDFILE command is not hard-wired to your userid, but will use whatever userid is logged on as the target on the receiving system.

If you are used to CLISTs but not familiar with REXX, you will probably find it strange. In fact, when I first tried to decipher a REXX procedure, I coined the expression, ''REXX is sure a dog.'' It seemed to coincide well with a television commercial airing at the time that starred Rex, the dog who liked using the sponsor's telephones.

It did not take me long when writing my own REXX procedures before I could see some distinct coding advantages over CLISTs. As of this writing, IBM has selected REXX over CLIST as the SAA procedure language. NetView Release 3 and TSO/E Version 2 are planned to include REXX.

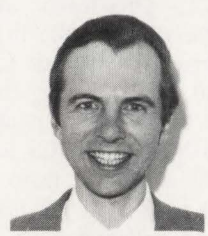
From personal experience, I can say that it would have taken me less time to convert 80 CLISTs from NCCF to NetView Release 3 REXX, than it did to NetView Release 2 CLIST. Just one IBM change caused a lot of work: NCCF processed messages one at a time, while NetView stores all messages it has received to date in an array.

Help

Want to learn more about ISPF? Just hit PF1, the Help key, once or twice and ISPF will try to display the most relevant Help panel wherever you are.

With ISPF 2.3 comes a completely new set of manuals. Not only are they more pleasant to read because of vastly improved design and print quality (the edit manual even has convenient section tabs), but also much of the material is rewritten.

ABOUT THE AUTHOR



Jon E. Pearkins is president of Certified Software Specialists Ltd., a Canadian company with a worldwide customer base for its line of IBM mainframe systems software. Certified Software, 54015 Range Road 212, Ardrossan, Alberta, Canada, T0B 0E0, (403) 998-0607.

Version 2.1 of DB2, which is now being installed in many sites, includes support for referential integrity. While this is a boon for most organizations, there are pitfalls that must be avoided.

Referential Integrity In DB2

By Lawrence Stevens

When Adam and Eve ate the forbidden fruit, they traded a life of blissful innocence for one in which a multiplicity of choices and decisions weighed upon them at every step. They were given the freedom to roam where they would, but their success or failure depended on the sweat of their brows and the wisdom of their planning.

The same analogy could be applied to relational databases in IS departments. Hierarchical systems were structured and inflexible, but they were safe and protected. Subtables were connected to higher level tables with intractable links. Relational databases like IBM's DB2 give users the freedom to customize the DBMS for the corporation by creating links as complex as the business environment requires. They also allow the freedom to make wrong choices which may cause data to lose integrity or strangle the organization with inflexible procedures.

Advantages Of Referential Integrity

Including referential integrity in Version 2.1 of DB2 is IBM's answer to users who want the adaptability of a relational system along with protection from stray data. Referential integrity allows IS to cement the links that bind the business aspects of an organization. It can mirror business procedures by linking such things as projects, customers, orders and commission records. It constrains the user from entering a foreign key value unless that same value exists as a primary key in the related table. This bars the user, for example, from entering a purchase order that includes a customer number unless that customer number actually exists in the customer table.

Relational integrity also prevents deleting a record without dealing in some way with all the dependent records. In this regard, DB2 allows you to work in one of three implementations. CASCADE deletes all the dependent records; SET NULL changes all the dependent records to NULL; and PROHIBIT disallows users from deleting records until all the dependent records are also deleted or renumbered.

Like all programming projects, there are efficient and inefficient ways to implement referential integrity in DB2.

Diane Brown, system consultant at Associated Insurance Companies (Indianapolis, IN), is working with her company's IS department to implement relational integrity because it raises the confidence level of her staff. She says, "Without referential integrity, most people are really nervous about doing an UPDATE. Once we have all the referential integrity rules in place and we are sure that they work, we will be able to relax and let the database do the worrying for us."

While referential integrity may reduce end-user worries about corrupting data once it is installed, implementing DB2 carries with it a responsibility. Incorporating correctly the organizational procedures while allowing enough flexibility for database users to have the freedom to account for exceptions is that responsibility. Because of this, experts are advising IS to take the time to study the business procedures, meet with administrators and users and test referential integrity constraints at every step of the implementation process.

Says Shaku Atre, president of Atre/Computer Assistance (Rye, NY), "Never before did IS have to be so aware of the day-to-day procedures of the organization. Referential integrity will bring the business and IS areas closely together and for many organizations that's a change in culture that has to be accomplished slowly and carefully."

Terry Mason, database project manager for the Bank of Montreal, agrees, "The hierarchical systems that many of us started with were, in a sense, outside of the organizational structure of the corporation. DB2 allowed us to make the database a reflection of the corporation and referential integrity will let us protect the database with the same rules and procedures that protect non-computer transactions. Referential integrity will make more

Succeed

PLATINUM *technology, inc.*
The DB2 Company™

PLATINUM *technology, inc.* is the only full service software company specializing exclusively in DB2. All our effort and energy is focused on delivering the broadest array of quality solutions for the DB2 user. As a result, IBM® has designated PLATINUM a Business Partner through the Authorized Application Specialist program specifically for DB2.

PLATINUM PRODUCTS

PLATINUM offers a complete line of software tools, education, and published products to ensure your success with DB2.

Software Products

The PLATINUM product portfolio consists of a complete family of administration, development, and end user DB2 software products. All are compatible with DB2 V1.3 & V2.1. The software products include:

- **RC/Query™** - A comprehensive DB2 catalog query tool.
- **RC/Update™** - The industry's best DB2 object management and data editing tool.
- **RC/Migrator™** - A complete object and data migration tool.
- **RC/Secure™** - An extensive DB2 security management tool.
- **PLATINUM Database Analyzer™** - The DB2 database and DASD analysis, audit, and management tool.
- **PLATINUM Report Facility™** - The DB2 query and reporting system for developers and end users operating in TSO and CICS environments.

DB2 Education Courses

A complete series of hands-on DB2 training courses. PLATINUM courses cover all aspects of DB2, QMF, and CSP. All courses are available either at your facility or at our Corporate Education Center.

- Introduction to DB2
- DB2/SQL Application Programming
- DB2 Application Planning and Database Design
- DB2 Database and System Administration
- Using DB2 and QMF
- CSP Application Programming

Published Products

The most recognized and authoritative DB2 standards, methods, and guidelines for DB2 implementation.

- **PLATINUM DB2 Guide/Online™** - The industry's leading standards manual for design, development, and administration of DB2 systems.
- **The PLATINUM Reference™ for DB2** - The quick, pocket-sized reference for DB2 information.

Support

All products and services carry our PLATINUM Quality Assurance Guarantee. Support is available 24 hours a day, 7 days a week via our toll-free hot line.

WORLDWIDE AVAILABILITY

PLATINUM's products and services are available around the world through our Affiliate Network. PLATINUM's full service capabilities include local support, education, and superior DB2 professional services.

with DB2!

PLATINUM *technology, inc.*
555 WatersEdge Drive
Lombard, IL 60148-9930
(312) 620-5000 FAX (312) 953-1923

For further information, in-house demonstration, or our exclusive no-obligation product evaluation call:

1-800-442-6861

CIRCLE #7 on Reader Service Card ▲

IBM Business Partner

work for the database administrator since he or she will have to resolve the relationships between the tables. But that's something he always had to do with hierarchical systems anyway.''

Application-Based Referential Integrity

While referential integrity is new to DB2 itself, most sites have had a version of it embedded in the application. Says Paul Tatro, regional sales manager for Platinum Technology, Inc. (Lombard, IL), ''Before DB2 had referential integrity, few IS departments or third-party software developers would be reckless enough to create a DB2 application without incorporating referential integrity constraints in the application itself.''

Tatro adds that there are advantages to having referential integrity in the DBMS rather than in the application. For one thing, DB2-based referential integrity would incorporate the actual organizational rules rather than the rules created for the application by a third-party developer. It also normally improves application performance because if referential integrity is in the DBMS, the application needs fewer lines of code and requires fewer transfers of data between the application and the DBMS. And finally, future development, as well as recovery work, is less complex because once implemented, referential integrity will not have to be considered each time a new application is created. Says Tatro, ''When referential integrity is included in the database itself, you write it once (it may take a while); then you don't have to ever worry about it again.''

Retrofitting Referential Integrity

That is the crux of the matter. Referential integrity is written once and for all in the DBMS and, as a result, applications are more efficient. Also, all applications used by the organization, even if they are written by third-party developers, are uniform as to business procedures. And someday, when all applications are new and have been written after referential integrity has been implemented, that will be true. For now, IS managers have to figure out how to retrofit referential integrity constraints without adversely affecting the application software.

Mike Giovinazzo, manager at Polaris Consulting Services Ltd. (Toronto, ON, Canada) warns, ''When placing referential constraints in DB2 that will affect applications that have their own constraints, you're opening up a can of worms. You can easily run into a situation where the DBMS and the application work against each other and you're going to get a screen load of error messages.''

Giovinazzo points out that even if you are careful to keep the DB2 version of referential integrity constraints consistent with the constraints in the application, there may be a problem of sequence. ''If an application were programmed so that it checks a customer table and DB2 first looks at the project table, your database might get out of sync. You have to study the application code carefully,'' he recommends.

Tatro agrees, ''If you're not familiar with the application code, make no assumptions. There's nothing stranger than some of the things running in the background of an application.''

Computer sites will benefit from using referential integrity properly.

When recreating in DB2 the referential constraints that are in the application, it is important to look for ''hidden'' links that may not be so obvious. Giovinazzo gives the example of an application in which the customer code includes within its number (say the last two digits) an indication of the location of that customer. That location code may be used as a referential check elsewhere in the application, but it could easily be overlooked when trying to replicate the application referential integrity with referential integrity in DB2. To avoid this problem, Giovinazzo advises first removing any such references against data to rows within the same table.

Implementing Referential Integrity

Since implementing referential integrity in the DBMS is a big job, it should be broken down into steps according to Atre. She recommends that her clients start implementing constraints that reflect those business rules that are clear and evident. Then they can continue to add new rules and expand on existing ones as new situations arise. She says, ''Begin by identifying 50 of the most important business procedures and add a few more every few months. It's the old 80/20 principle. Twenty percent of the procedures are used 80 percent of the time. It's the most-used 20 percent that you should work on initially.''

Atre also points out that many rules are fuzzy and it is difficult to encode the ambiguities and exceptions that most corporations feel are necessary if they are to function in a flexible manner. She believes that her approach of implementing referential integrity slowly and in steps will eliminate the danger of creating an inflexible system.

Tatro agrees and adds that one problem some DBAs experience is that they look at the rule book instead of how the organization really works. He says, ''When you question users to find out how their operation works, be sure to ask them about exceptions. Even if they say they always have a customer number before they process an order, make sure they really mean *always* and not 90 percent of the time. If there are exceptions, make room for them. Provide a customer number for all customers waiting to be assigned a number or teach people how to use the NULL function.''

Learning about the business aspects of a company is a long process involving the software engineer, administrators and end users. Up until now, since DB2 did not have referential integrity, the data analysis period involved finding the ways that users will want to access the DBMS. Now DBAs have to learn how users should *not* be able to access the DBMS. This complicated process is often a drain on IS departments that have to postpone other projects. So it is tempting to decide not to bother with referential integrity. Says Atre, ''Of course, it's always a judgment call as to whether it is time-efficient to incorporate referential integrity constraints into DB2. But in most cases, for evey hour you spend up front, you'll save tens of hours later.''

One way to help amortize the time spent in creating referential integrity constraints is to use the process to clean up general business procedures. Atre advises, ''Speak to users and try to get a consensus not

See DB2 *page 95*

Who are you going to turn down?

Buy the DB2 performance monitor that meets *all* their needs.

DBAs, System Programmers, Application Developers, Auditors. They all know how critical their jobs are to the performance of your DB2 system. And each group needs a unique type of performance information to optimize productivity. That's why Insight/DB2 has user-specific menus for all four groups, designed to provide the performance information they need most.

And Insight/DB2 offers much more. It's the only DB2 performance monitor to offer both batch and online reporting. You get both current and historical data, access to all DB2 data, multiple output options for the performance data, and Insight/DB2's unique reporting language that lets your users write their own performance reports. Insight/DB2 will meet all of their performance needs, both now and as your installation matures.

Insight/DB2: The Complete Monitor for DB2.

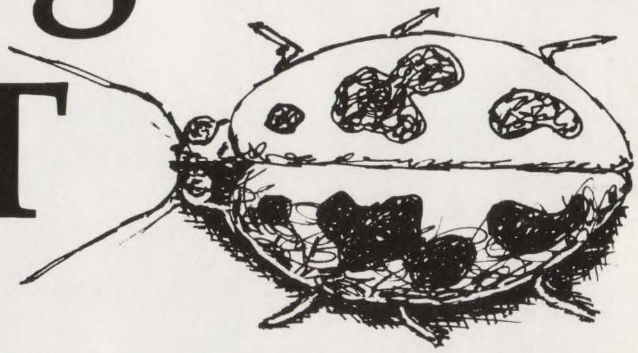
For more information or to start your free 30-day trial of Insight/DB2, call Goal Systems at **(800) 422-5849**.

CIRCLE #79 on Reader Service Card ▲

Goal Systems

Goal Systems International Inc. • 7965 N. High Street • Columbus, Ohio 43235 • Phone: (800) 422-5849
Goal Systems International S.A.R.L. • 88 avenue de Wagram • 75017 Paris, France • Phone: (1) 42 67 55 55 • Telex: 641.094

Debugging With COBTEST



By Alan Friend

It amazes me how few COBOL programmers, especially those writing batch systems, use interactive debugging facilities. I have heard the following reasons for this.

- Debugging facilities use too much system resources.
- I write very small modules which are easy to debug.
- READY TRACE and DISPLAY do the job just fine.
- I'm not familiar with any interactive debugging facilities for batch COBOL.
- How can you interact with a batch program, anyway?

While some of these may be valid points, they also seem to reveal a general lack of familiarity with the use and benefits of interactive debugging. When the name of the game is programmer productivity, it makes sense to utilize facilities which speed up the testing and debugging process. The reason for this article is to discuss the capabilities of COBTEST, IBM's VS COBOL II interactive debugging tool.

A Little History

Initially, IBM provided OS COBOL Interactive Debug, consisting of the TESTCOB command and its subcommands, for programs compiled under the OS/VS COBOL and OS Full American National Standard COBOL, Version 4 compilers. Commonly called TESTCOB by programmers, this tool enabled programmers to monitor and interact with programs by entering line commands from the terminal while the program executed. But few programmers took advantage of TESTCOB, citing the reasons mentioned at the start of this article. I suspect their reluctance arose out of unfamiliarity. Many old-time COBOL programmers either originally learned their COBOL in schools which did not offer instruction in this sophisticated tool or worked in shops which did not encourage TESTCOB's use because of limited resources.

When IBM introduced Release 1 of VS COBOL II, they provided the VS COBOL II Debug Tool. This debugging facility, often referred to as COBTEST by programmers, enabled the programmer to debug and test interactively by entering line commands at the terminal or to debug by executing the program under COBTEST in batch mode.

With Release 2 of VS COBOL II, the Debug Tool was now officially called COBTEST in the IBM manuals. It was greatly enhanced to allow testing and debugging interactively in full-screen mode as well as in batch and line modes. It is this later release of COBTEST that is presented in this article.

Basic COBTEST Tasks

When testing or debugging a program, it is helpful to be able to trace the logical flow of the program, list the contents of selective identifiers (for example, data item or file name) at specific points in the execution and make changes to the values of identifiers to see what the effect will be. OS/VS COBOL programmers often coded READY TRACE and RESET TRACE to follow the flow of control in a program; although the COBOL II compiler checks the syntax when these statements are coded, it ignores them during program execution. OS/VS COBOL programmers also made extensive use of the DISPLAY (and sometimes EXHIBIT) verb to check the contents of identifiers at specific points in the program. They sometimes coded conditional statements to change the contents of identifiers when certain testing conditions were met during program execution. COBOL II programmers may still use these latter techniques, although DISPLAY must be used instead of EXHIBIT since EXHIBIT is not available in COBOL II.

How much easier it is, however, to use COBTEST facilities to carry out these tasks! COBTEST allows you to suspend execution of the program at any COBOL verb, list the values and attributes of identifiers and the status of files, alter the values of identifiers at any time and trace the logical flow of the program. The beauty of this is that these tasks are done without coding them in the program and, therefore, without the need to recompile the program when the "test" code needs to be changed or removed. You merely specify the appropriate COBTEST commands in an input dataset for batch mode or at the terminal for the interactive modes and run the program.

Some caveats need to be mentioned. The TEST compiler option must be in effect in order to use COBTEST. Because this option reduces program efficiency, the program should be recompiled without TEST when debugging is completed. Also, although the output from COBTEST commands is essentially the same whether batch mode or one of the interactive modes is used, it is not possible to interrupt the program or interact with it at the terminal once execution of the program under COBTEST in batch mode has begun.

The Three COBTEST Modes

Batch mode can be used for debugging batch and CICS programs. (CICS programs cannot be run under the interactive modes.) It allows you to enter debugging commands in a dataset, with the commands being executed as soon as they are read in by COBTEST. Obviously, it is not possible to make any changes to this command list since it is run as a batch job. The output from the COBTEST commands will be sent to a dataset that can

be viewed either when the job terminates successfully or abends. Batch mode uses less resources than the interactive mode. You may wish to use it when you do not want to tie up your terminal for long periods of time or when you wish to use COBTEST facilities to gather information about program execution for later review.

In interactive line mode you enter debug tool commands sequentially at the terminal, one line at a time. Output from the commands is also displayed at the terminal sequentially. Use of COBTEST in line mode is intended primarily for programmers who only have access to a teletype terminal or for installations which do not have ISPF. A disadvantage of line mode is that source code cannot be viewed at the terminal.

Interactive full-screen mode is the most powerful of the three COBTEST modes. To use it, ISPF Version 2 under MVS-TSO, MVS/XA-TSO, MVS/ESA-TSO OR VM-CMS must be installed. The screen can be divided into three windows of varying size and shape, displaying the source code in one, a log area in another and the contents of program variables in the third. Full-screen mode allows you to dynamically step through the program, watching the source code scroll by as it executes with the executing line of code highlighted. While this is occurring, you can view the log area to see the output of COBTEST commands as they execute and monitor the values of program variables in an auto monitoring area as they change. If PDF is installed, you can also split the physical screen into two logical screens in order to do other tasks. Thus, you can edit a dataset in one logical screen while remaining in COBTEST in the other.

You should now have a feel for the capabilities of the three COBTEST modes. The remainder of this article will expand upon the uses of COBTEST.

More About COBTEST Tasks And Commands

Figure 1 contains an alphabetical list of all COBTEST commands. It indicates in which mode each command can be used and gives a capsule explanation of its use. Further information, including complete syntax and usage examples, can be found in the following:

- The IBM manual VS COBOL II Application Programming: Debugging, Release 2
- Line mode by entering HELP or HELP command-name at the terminal
- Full-screen mode by entering HELP on the command line or hitting the PF key defined for the help function; a Main Help Menu will then be presented from which a topic or command can be selected by number.

Figure 2 shows the commands that would be used to perform the various COBTEST tasks.

A sample debugging session, in which most of the common COBTEST tasks and commands are used, is shown in the section below.

FIGURE 1

COMMAND	MODE	PURPOSE
AT	All Modes	Set break points
AUTO	Full-Screen	List variables in auto-monitoring area
COBTEST	Batch, Line	Invoke the COBOL II debug tool
COLOR	Full-Screen	Set color and intensity attributes
COMMENT	All Modes	Insert comments into COBTEST input and output
DROP	All Modes	Delete symbols created with EQUATE command
DUMP	All Modes	Terminate debugging and give a snap dump
EQUATE	All Modes	Establish symbols for program data-names
FLOW	All Modes	Trace control flow of selected portions of program
FREQ	All Modes	Tally verb use during program execution
GO	All Modes	Start or resume execution, recognizing breakpoints
HELP	Line, Full-Screen	Provide information about COBTEST commands
IF	All Modes	Cause conditional execution of COBTEST commands
LINK	All Modes	Simulate a non-existent calling program
LIST	All Modes	Display attributes and values of selected storage
LISTBRKS	All Modes	List current breakpoints
LISTEQ	All Modes	List active symbols created by EQUATE command
LISTFREQ	All Modes	List tally of verb usage if FREQ was turned on
LISTINGS	Full-Screen	Make associations between source listings and debuggable program units
MOVECURS	Full-Screen	Move cursor between last position in source or auto-monitoring area and command line
NEXT	All Modes	Set temporary breakpoint at next verb to be executed
OFF	All Modes	Remove AT breakpoints
OFFWN	All Modes	Remove WHEN breakpoints
ONABEND	All Modes	Permit execution of a command list if program abends
PEEK	Full-Screen	Allow viewing of line number obscured by AT breakpoint feedback
POSITION	Full-Screen	Cause specified line in source, auto-monitoring or log area to move to top of its area

Continued on Next Page

A Sample COBTEST Session

Assume that the program you have just written has abended with an S0C7. The purpose of the program is to read a record, process it and write a detail line. A subprogram that has not yet been written is to be called after all record processing has been completed. The type of processing will be determined by a parameter passed from a non-COBOL program. For testing purposes, your input dataset contains only 50 records. See Figure 3 for program excerpts.

Assume you have defined all files needed by the program with the output files allocated to the terminal and have entered interactive full-screen COBTEST mode in order to quickly solve the abend. You proceed as follows.

First, assuming the subprogram is called SUBPROG, you enter

PROC SUBPROG

to trap calls to the non-existent submodule.

It is not necessary to actually execute the calling program. You can enter the command which establishes addressability to the 01 identifier INPARM in the LINKAGE SECTION that will receive the parameter from the non-COBOL calling program. The command is

LINK INPARM

You can now enter the command

SET INPARM = 'B'

to simulate the passing of the parm value 'B' to INPARM.

Entering either

GO or RUN

begins execution of your program.

The program abends. COBTEST indicates a system abend code of 0C7 and displays the line number of the statement that was executing, line 11400. Looking at the source code, you find that the COBOL statement is

IF DETAIL-LINE-CNT
GREATER THAN +55

Since an 0C7 is a data exception, you suspect that there is invalid data in DETAIL-LINE-CNT. You may want to enter the command

EQUATE L-C DETAIL-LINE-CNT

to permit you to enter L-C instead of DETAIL-LINE-CNT at the terminal (this saves a few keystrokes).

To view information about DETAIL-LINE-CNT, you now enter

LIST L-C or LIST DETAIL-LINE-CNT

COBTEST indicates that this is a PIC S99 COMP-3 field and displays the contents in hex format as F0F0. COBTEST specifies that this is invalid data for this data type; the data is not packed.

To determine why DETAIL-LINE-CNT originally contained unpacked zeros, you must examine the source code. You can place the cursor in the source area and enter

SEARCH DETAIL-LINE-CNT

to locate the character string DETAIL-LINE-CNT. You find that DETAIL-LINE-CNT is defined as an elementary item within the COMP-3 group item COUNTERS. Entering

SEARCH COUNTERS

reveals that you coded

MOVE ZERO TO COUNTERS

at line 10600. A group move is treated as an alphanumeric move; hence, DETAIL-LINE-CNT was improperly initialized to unpacked zero.

To change the contents of DETAIL-LINE-CNT to packed zero, you enter

SET L-C = 0

Entering the LIST command again verifies that the contents of the field are properly packed.

You enter similar SET commands for the other elementary items in COUNTERS.

You now enter

GO

to resume execution from the same state-

PREVDISP	Full-Screen	Allow redisplay of a previous panel displayed by program if ISPF Services were used
PRINTDD	All Modes	Specify file where output from RECORD or commands with PRINT option will be directed
PROC	All Modes	Trap calls to a non-existent subprogram
PROFILE	Full-Screen	Allow setting of COBTEST session parameters
QUALIFY	All Modes	Change qualification of names or line numbers used in COBTEST commands
QUIT	All Modes	Terminate debugging session
RECORD	All Modes	Activate or deactivate logging of COBTEST session
RESTART	Line, Full-Screen	Load new copy of program and begin execution
RESTORE	Full-Screen	Restore source area to last point of execution
RUN	All Modes	Execute program without interruption caused by active breakpoints
SEARCH	Full-Screen	Search source, log or auto-monitoring area for a given character string
SELECT	Full-Screen	Allow viewing of frequency count on a line with multiple statements
SET	All Modes	Initialize or change values of program variables
SOURCE	Full-Screen	Open, close or change size of source area
STEP	All Modes	Cause execution of a specific number of program statements
SUFFIX	Full-Screen	Open or close suffix area within the source area
SYSCMD	Line, Full-Screen	Allow use of CMS commands while in debugging session
TRACE	All Modes	Display a flow of program execution
VTRACE	Full-Screen	Allow dynamic viewing of program execution
WHEN	All Modes	Set up continuous monitoring of a condition before each verb in program
WHERE	All Modes	Show where execution has been suspended

F I G U R E 2

TASK	COMMANDS
Setting Breakpoints	AT NEXT ONABEND PROC WHEN
Removing Breakpoints	OFF OFFWN
Resuming Program Execution	GO RESTART RUN
Analyzing Program Behavior and Flow	FLOW FREQ STEP TRACE VTRACE
Testing Program Conditions	IF WHEN
Referring to Other COBOL II Programs	QUALIFY
Simulating Non-Existent Calling or Subprograms	LINK PROC
Setting Values of Variables	SET
Displaying Values of Variables	LIST
Creating and Deleting Alias Names	EQUATE DROP
Determining Status of COBTEST Settings	LISTBRKS LISTEQ LISTFREQ WHERE
Directing COBTEST Output to Datasets	PRINTDD
Logging COBTEST Session	RECORD
Terminating COBTEST Session	DUMP QUIT

ment that previously caused the abend. The call to SUBPROG is trapped by the PROC command and you enter

GO

again to continue. Since DETAIL-LINE-CNT and the other fields within COUNTERS now contain valid data, the program ends normally.

You realize you must change your program's code at line 10600, perhaps to

INITIALIZE COUNTERS

then recompile and relink, then run the program through COBTEST again to verify your finding.

But wait! The program output displayed at the terminal is incorrect! The detail lines contain data that should only be written when the passed parameter is 'A'. However, you specified a value of 'B' for INPARM. Also, the last record was written twice.

You decide to re-execute the program with the TRACE command to check the control flow of the program as it executes. Entering the command

TRACE NAME

will give a trace showing the line number of each paragraph entry point along with the paragraph name, as well as the line number of the first statement beginning each conditional group of statements within each paragraph.

Before entering the RESTART command, you need to set breakpoints so you can halt execution at the beginning of the program and set the values of INPARM, DETAIL-LINE-CNT and the other elementary fields in COUNTERS to their desired values. This is because RESTART loads a new copy of your program, thereby reinitializing program variables. All breakpoints and other COBTEST settings

COBTEST allows testing and debugging interactively in full-screen, batch and line modes.

will remain in effect.

You therefore enter

AT 10400

to halt execution at the first executable instruction in the PROCEDURE DIVISION.

You now enter

RESTART

When the new copy of the program halts at line 10400, you enter the commands

SET INPARM = 'B'
FREQ
AT 10600 (NEXT)

The FREQ command turns on tallying of verbs; this may be useful to determine how many READS and WRITES have been executed.

Since you found that the COBOL statement at line 10600 moved alphanumeric zero to DETAIL-LINE-CNT, you enter the AT statement with the subcommand NEXT to halt execution at the statement following the COBOL statement. This will allow you to reset DETAIL-LINE-CNT and the other items in COUNTERS to properly packed zero.

You now enter

GO

twice in succession. When execution halts at the statement following line 10600, you enter

SET L-C = 0

as well as the other necessary SET commands for the elementary items in COUNTERS.

You then enter

GO

to continue execution until the PROC SUBPROG command interrupts the program at the COBOL call statement and

GO

again to proceed to end-of-job.

Since TRACE NAME has been in effect, you can view the log area to see the flow of execution of the program. You learn that the program never executed the paragraph that should have been performed if INPARM was 'B'. Instead, the paragraph should only have been performed when INPARM had a value of 'A' was performed. You find that the EVALUATE statement in the program was coded incorrectly. The trace reveals that the proper WHEN clause at line 12100 was executed, but that it caused a branch to the wrong paragraph because you coded the wrong paragraph name at line 12200 in error:

PERFORM 0100-PARA-A

instead of

PERFORM 0200-PARA-B

You enter the COBTEST command

LISTFREQ

and find that, as you suspected, 51 READ and 51 WRITE statements were executed by the program. This means a program logic error since your input dataset contains only 50 records. Examining the code reveals that you are executing one too many READs and have read beyond end-of-file. You then write the 50th output record an extra time. To verify this you restart the program as described above and use the commands

AT 11100 COUNT (50,1)
GO

to set a breakpoint at the beginning of the in-line perform that controls reading and writing of records the 50th time this procedure is reached and every time thereafter. Now enter

GO

The READ will be executed 49 times and execution will be interrupted before it is executed the 50th time.

When execution halts at line 11100, you enter

F I G U R E 3

```
                  .
                  .
008000  WORKING-STORAGE SECTION.
008100      SKIP1
008200  01 COUNTERS         COMP-3.
008300      05 DETAIL-LINE-CNT        PIC S99.
008400      05 A-RECORDS-WRITTEN      PIC S9(5).
008500      05 B-RECORDS-WRITTEN      PIC S9(5).
                  .
                  .
008900  LINKAGE SECTION.
009000      SKIP1
009100  01 INPARM                     PIC X.
009200      EJECT
009300 PROCEDURE DIVISION USING INPARM.
009400      SKIP1
009500*****************************************************************
009600*     MAINLINE CONTROL PARAGRAPH - HOUSE-KEEPING, READ EACH        *
009700*     RECORD AND PROCESS BASED ON VALUE IN INPARM RECEIVED FROM    *
009800*     CALLING PROGRAM.     AT END OF INPUT FILE, CALL 'SUBPROG'    *
009900*     TO DO SUMMARY REPORT PROCESSING.                             *
010000*****************************************************************
010100      SKIP1
010200 0000-MAINLINE.
010300      SKIP1
010400      OPEN INPUT  INFILE
010500           OUTPUT OUTFILE
010600      MOVE ZERO TO COUNTERS
                  .
                  .
011000      PERFORM    UNTIL INFILE-EOF
011100         READ INFILE
011200              AT END SET INFILE-EOF TO TRUE
011300         END-READ
011400         IF DETAIL-LINE-CNT GREATER THAN +55
011500            MOVE ZERO TO DETAIL-LINE-CNT
011600            PERFORM 0400-HEADER-RTN
011700         END-IF
011800         EVALUATE INPARM
011900            WHEN     'A'
012000               PERFORM 0100-PARA-A
012100            WHEN     'B'
012200               PERFORM 0100-PARA-A
012300         END-EVALUATE
012400      END-PERFORM
012500      CALL 'SUBPROG'
012600      CLOSE INFILE
012700            OUTFILE
012800      GOBACK.
012900      EJECT
013000*****************************************************************
013100*     0100-PARA-A AND 0200-PARA-B, PROCESS AND WRITE THE OUTPUT    *
013200*     RECORD.  PROCESSING DEPENDS ON THE RESOLUTION OF EVALUATE    *
013300*     STATEMENT IN 0000-MAINLINE PARAGRAPH:                        *
013400*            IF INPARM IS 'A', 0100-PARA-A IS PERFORMED            *
013500*            IF INPARM IS 'B', 0200-PARA-B IS PERFORMED            *
013600*****************************************************************
013700      SKIP1
013800 0100-PARA-A.
                  .
                  .
014800 0200-PARA-B.
                  .
                  .
```

STEP

repeatedly to step through the procedure one statement at a time. Following the flow of execution in this paragraph, you find that

READ INFILE
AT END SET INFILE-EOF TO TRUE

was coded at the beginning of the procedure instead of at the end. Further examination of your source code reveals that there was no initial READ statement and that you coded the PERFORM in question UNTIL INFILE-EOF.

Thus, you have been able to isolate several problems in your COBOL program in one COBTEST session. The next step would be to make the necessary corrections to the source module, recompile and relink, then execute the module again to test the changes.

Conclusion

The preceding example represents only one possible solution using various commands I chose to debug a sample program. Hopefully, it demonstrates the power and flexibility of the COBTEST tool and convinces at least a few COBOL II programmers to enlist COBTEST during the program development phase of their next project.

ABOUT THE AUTHOR

Alan Friend is currently a senior data processing training associate at New York Life Insurance Co. He has worked on COBOL II systems and is responsible for training his company's data processing personnel in COBOL and COBOL-related areas. (212) 576-4409.

DB2 *from page 88*

only on the present data usage requirement, but also on ways to improve procedures.''

Efficient Coding

Like all programming projects, there are efficient and inefficient ways to implement referential integrity in DB2. Most experts recommend doing performance checks before and after a referential integrity constraint is implemented. If performance degrades more than is acceptable, more efficient referential integrity procedures might help.

Giovinazzo gives the example of trying to implement referential integrity constraints on a customer table and customer order table. If the primary key on the customer table were customer ID and there was an index on the customer order table in which the customer ID is the most significant field, referential integrity would be easy to implement and low on overhead.

On the other hand, if the customer order table contains the order ID as the most significant field, then when DB2 was enforcing the constraint it would use a lot of resources because it would have to process more rows in the customer order table. In that case Giovinazzo recommends creating a new index on the customer order table that has the customer ID as the most significant field. Giovinazzo also recommends establishing indexes on the foreign key tables to eliminate the need for DB2 to scan all the data in the foreign key table.

However, even with the most efficient coding, referential integrity may create some decrease in performance. It may also provide constraints which users are not used to. Therefore, there may be some user resistance to it. When selling referential integrity to users, it is important to include not only the obvious benefits, but also those benefits which are not so obvious. Atre points out that future applications will have reduced coding needs and will, therefore, be more efficient. Future development work will also often be faster since the referential integrity already existing will not have to be altered.

''In the long run,'' she asserts, ''almost every computer site will come out ahead if it makes good use of DB2's referential integrity capability.''

ABOUT THE AUTHOR

Lawrence Stevens is a free-lance writer based in Springfield, MA.

A Perspective On Performance Management

By Reba L. Chaisson

Performance Management (PM) encompasses monitoring and tuning the operating system. The objective is to minimize service time, enhance throughput and extend the life of the system. It consists of making optimal use of the computer's resources, as well as assessing the impact of workload, hardware and software changes.

PM also comprises implementation and evaluation of various I/O devices, memory, operating systems, software tools and monitors. Performance analysts design paging and I/O configurations, as well as perform bottleneck, workload and trend analyses.

Prior to the advent of PM, the recommended solutions to poor throughput and service times were hardware solutions — memory or CPU upgrades costing from hundreds of thousands to several million dollars. At that time, management was unaware of less expensive alternatives (that is, parameter tuning, I/O reconfigurations, redesign of the paging subsystem). Now that management is better informed of the options, there has been an aggressive development of the PM specialization in many companies.

PM Skeptics

Some managers are skeptical about PM, however. Why? For one, upgrading is simpler and more straightforward. Historically, increasing CPU and memory capacity has accomplished the objective of improving performance and accommodating workload growth.

Secondly, management must be politically and financially committed to providing the tools that make PM an integral part of the data center. PM is a relatively new profession and learning to do the job well requires extensive training and subsequent experience.

Tools such as education, performance monitors, degradation analyzers and data collectors are necessary for a successful PM implementation. The education provides the knowledge base and analytical techniques necessary to perform analyses. Performance monitors and degradation analyzers provide windows to the system aiding in workload impact assessment. Finally, data collectors capture data for post analysis, historical trending and capacity planning.

Capacity planning is the specialty that forecasts the time and to what degree a workload's performance will change with the current or anticipated resource capacity. Capacity planners do this by projecting the historical data in conjunction with anticipated workload growth and system capacity. They use various sophisticated statistical and analytical techniques to determine the upgrade necessary to accommodate workload growth. There are modeling tools available on the market which take in historical data and system parameters and, subsequently, build a model of the system. With the model, workload changes (that is, growth) can be made for various points in time and a forecast charted. This type of analysis is beneficial to management because it acts as a budget aid by predicting at what point monies will be needed for upgrades.

Management's Focus

The political commitment enters on this note. Involving system performance, models and forecasts in the company's strategic planning requires a sharp adjustment to traditional management thinking. Rather than solely considering the direction the industry is going and being primarily concerned with remaining on the leading edge of technology, management will instead have to learn to focus on the technical and financial goals for its own company. Too often, the focal point of DP management has been with the industry. This has been dictated by a fear of being left behind or out in the cold. However, PM and capacity planning is the opportunity for management to regain control of its own company. These specializations place the focus back on the company and its goals and objectives. Effective PM and capacity planning opens the mind of management by making it aware of the many vendors, products and techniques available for resolving a conglomerate of technical issues. As for the fear of not remaining on the leading edge of technology, a bigger fear should be of *falling off of that edge*. Perhaps it is more astute to remain one or two steps behind.

Do not interpret this as meaning management should become oblivious to the direction the industry is taking. However, do understand that management's primary concern should be what is best for the company. If solid-state will solve the problem for a two-cylinder performance critical dataset with a 1:1 read-to-write ratio, then why invest in read-oriented cache? If the company cannot benefit from a newly announced operating system, then why upgrade in the first year of release? If there is a 16K segment caching algorithm available, then why purchase a device that caches at full track? Management's concern should be with *optimization*.

Cost Factor

The cost potential of maintaining a well educated, well informed and experienced PM staff can be expensive. Education, salary and benefits for a group of only three analysts would be in excess of $150,000 per year — this is exclusive of the aforementioned tools. In an effort to minimize this cost, many companies have their systems programmers double as performance analysts. The feasibility of this arrangement depends on the size and volatility of the company business. The larger and/or more dynamic the data center is, the less time the systems programmer will have to develop his/her performance expertise.

Systems programmers spend much of their time testing and implementing new software releases and maintaining and troubleshooting software problems. Coupling PM with these responsibilities may allow systems programmers only enough time to do bottleneck analysis — analysis after a problem has become evident. The objective of PM is to assess the impact of changes *prior* to their implementation and minimize or prevent any impact to the workload. Effective PM cannot be done without ongoing monitoring, tuning and workload analysis. Understanding the environment cannot happen without impact assessment of environmental changes each step of the way.

Performance consultants are a viable option for minimizing personnel costs. These specialists should have expertise in systems tuning, as well as excellent analytical, planning and communication skills.

Tailor-made

Despite the chosen alternative, the most economic and strategic manner in which to optimize resources and extend the life of the system is to monitor, tune and configure the system with the company's goals and objectives in mind. Instituting PM with the proper personnel and the necessary tools can save or postpone millions of dollars in hardware costs and restore control of the company from the industry to management.

ABOUT THE AUTHOR

Reba L. Chaisson is a partner at Operating Systems Performance, Glenwood, IL. Her background includes nine years of experience in data processing.

For information on advertised products, fill out the READER SERVICE CARD between pages 66 and 67.

The Illusion of Measurement

By H. Pat Artis

H. Pat Artis, President of Performance Associates, Inc.

For a few moments, consider an hypothesis. Today, we are rapidly losing our ability to effectively measure computer systems.Unfortunately, our awareness of this dilemma is limited by the illusion of computer measurement. Simply stated, it is easy to confuse the accessibility of measurement data we enjoy today with the availability of the data we really need. Unless we address this trend, our long-term ability to measure, tune, allocate costs and plan for systems may be compromised.

Twenty Years Of Effort

In reflection, it still seems hard to believe that 20-odd years have passed since the introduction of the first set of measurement tools for 360 class systems. In the late 1960s, we witnessed the introduction of software monitors, hardware monitors and SMF with OS/MVT Release 18.0. While software and hardware monitors were remarkable inventions, SMF was a response to the requests from hundreds of installations that wanted to understand the nature of and to account for resource consumption. While TCMs have for the most part eliminated the probe points on which mainframe hardware monitors depended, software monitors and the SMF log file still continue to flourish today.

During the early years, log file analysis was a black art involving manipulation of SMF record bytes, halfwords and offset fields in Assembler, PL/1 and a variety of other languages. Unfortunately, the complexity of the record formats and the time required to develop and update analysis programs for new SMF releases limited the number of sites that exploited the data. For those who chose to analyze the data, the majority of the time was spent decoding the contents of fields rather than interpreting their meanings. While this may not have been the most efficient use of time, the measurement zealots who pursued this path dealt with the data on a daily basis and continued to present IBM a wealth of requirements for addressing the holes in the SMF measurement scheme and MF/1 and RMF when they were introduced in the 1970s.

A Turning Point

Perhaps the single most important event in the first two decades of computer measurement was not the introduction of a new data source but the use of the SAS language to analyze the data. This approach, pioneered by Barry Merrill, led to the development of a variety of packages that decoded the SMF, RMF and other records recorded in the SMF log and stored them in an easily accessed and understood SAS data format. Today, thousands of sites employ MXG, MICS or some other SAS-based performance database as their primary vehicle for analyzing SMF, RMF, and other log file information. After installing one of these tools, users immediately find themselves almost overwhelmed by the available data--the legacy of almost 20 years of effort.

One unfortunate consequence of this bonanza of information is the illusion of measurement. That is, it is easy to confuse the accessibility we enjoy to measurement data with the availability of data we really need. Moreover, few of these new users ever question whether the underlying data sources really provide the data elements that they need.

What Do We Need?

Often, I hear questions like, "What kind of new capacity planning tools do we need to plan for today's complex environments?" While this question may have some merit, it might be better to ask, "What kind of data do we need to plan for today's environments and is it available?"

To illustrate these concerns, consider two measurement issues that we face today. Expanded storage was introduced several years ago with the initial 3090 processor family. Moreover, expanded storage is a cornerstone of MVS/ESA. Unfortunately, little if any measurement data is available for understanding the use of and managing this complex resource.

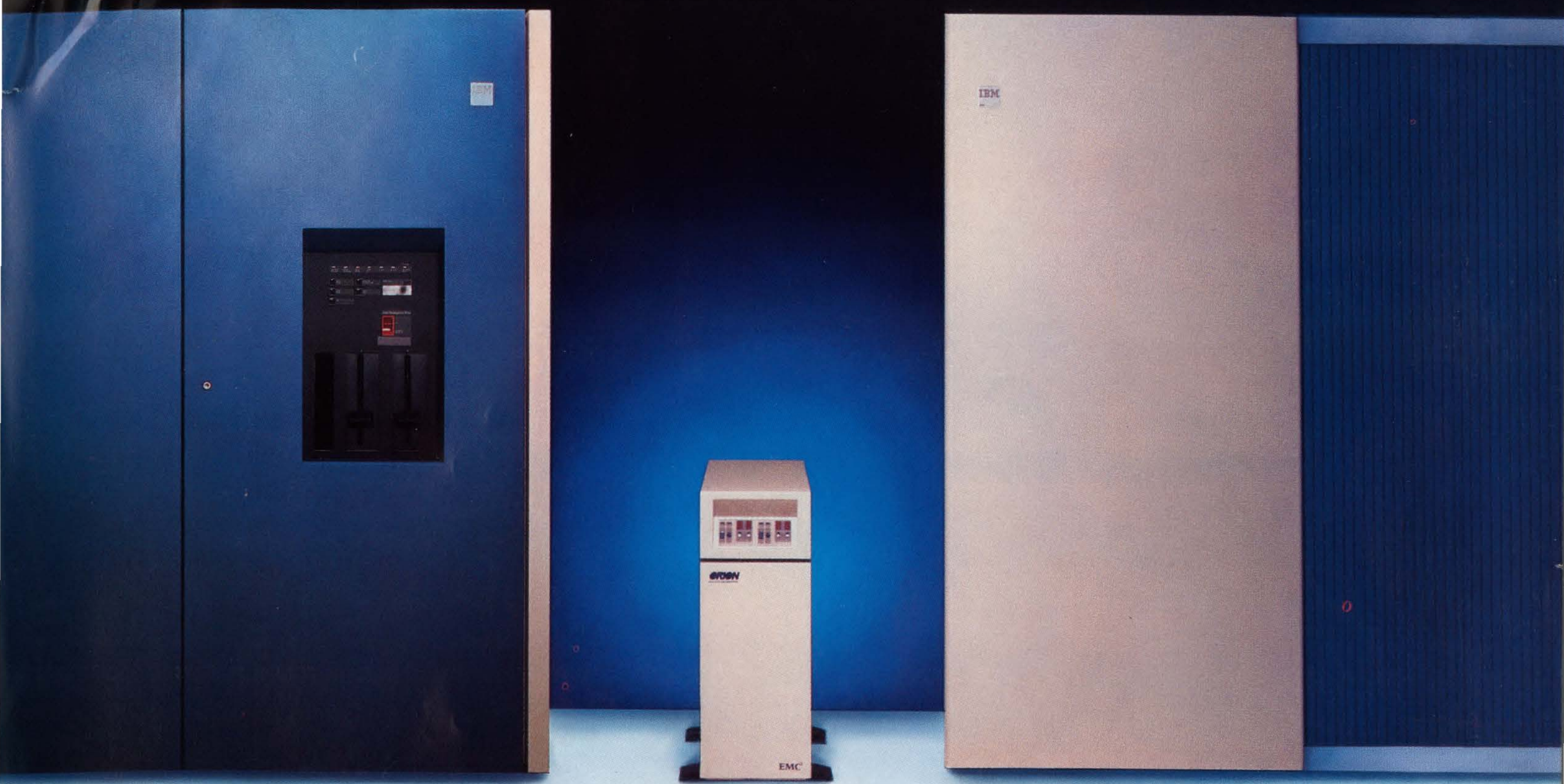
Cross memory services were introduced with MVS/SP 1.3 in the early '80s and have been significantly expanded in MVS/ESA. Yet, it is almost impossible to measure the resources that address spaces like DB2 consume. Moreover, IBM's INFO/SYS Q&A entry observed that DB2 bypasses many of the available I/O measurement interfaces for the sake of efficiency. This problem will be compounded by new applications like ImagePlus and the Repository Manager which are based on DB2.

While these problems are common to many installations today, it is often difficult to find a significant constituency of users to lobby for these requirements.

Dispelling The Illusion

To address this dilemma, we must focus beyond data we have available and clearly identify the data we actually need to measure, account and plan today's environments. Then we must communicate these requirements to IBM and other hardware and software vendors. Unless we address these issues, we will soon find that measurement data on which we depend is mostly an illusion and that our capacity planning and performance management methodologies are interesting relics of an era that has passed.

EMC's ORION:
The Cost-effective Solution to Your I/O Performance Problems.



EMC's ORION Solid State Disk Subsystem

Maximum Performance

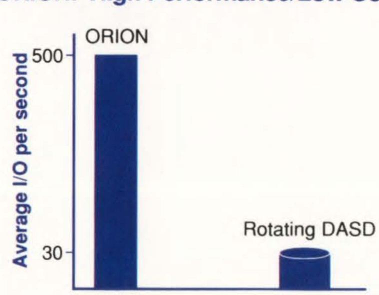
EMC's ORION is the fastest solid state disk subsystem available for your mainframe computer. ORION features a technologically advanced design enabling you to receive unprecedented performance gains.

An integrated 3880 type storage director, in addition to features inherent in solid state technology, gives ORION an access time of 0.1 millisecond — a performance milestone.

What's more, EMC protects your investment by making ORION compatible with all IBM 370 and IBM PCM computers. Therefore, performance boosts are realized well into the future, when CPU upgrades become necessary.

EMC's ORION — the I/O solution you have been waiting for to maximize your system's performance and productivity.

ORION: High Performance/Low Cost



Minimum Price

EMC's ORION Solid State Disk Subsystem is the lowest cost solid state solution to your I/O performance problems. Its state-of-the-art design gives you a substantial return on your investment for years to come.

ORION's small footprint and low power requirements eliminate costly computer room renovations. Ease of installation and low cost of ownership make ORION an expedient and economical solution to your I/O performance problems.

EMC's ORION — the most economical performance boost for your current and future mainframe systems.

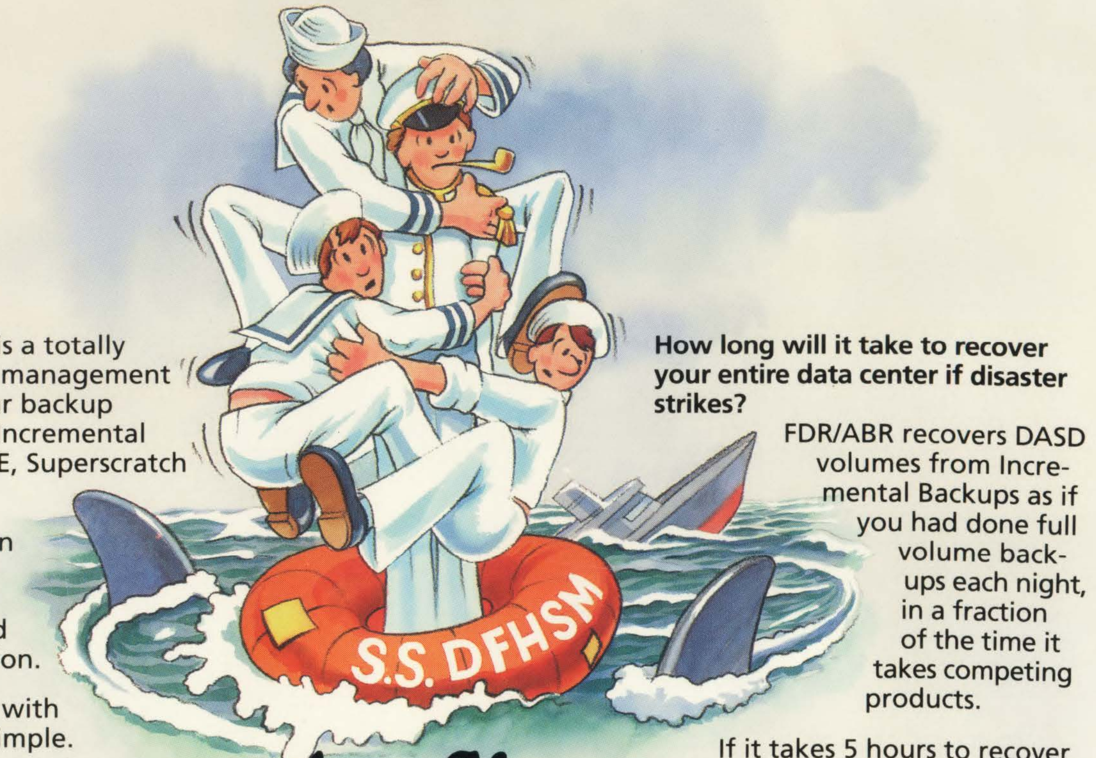
EMC²
The System Enhancement Company.

IBM is a registered trademark of International Business Machines Corp.

For more information, call:
1-800-222-EMC2
In Mass., call (508) 435-1000

CIRCLE #95 on Reader Service Card ▲

When Disaster Strikes Finding Your Backup Recovery Product Inadequate...



The FDR® system is a totally integrated DASD management system for all your backup needs, including Incremental Backups, ARCHIVE, Superscratch and reporting.

ABR®'s Application Backup System will backup the data sets required for each application.

Disaster recovery with FDR/ABR is very simple.

SAR (Stand Alone Restore) recovers your system volumes without an operating system.

Restore the ABR catalog and off you go. ABR, using backward recovery, will recreate DASD volumes from their most current backup. ABR will automatically call for the proper backup tapes.

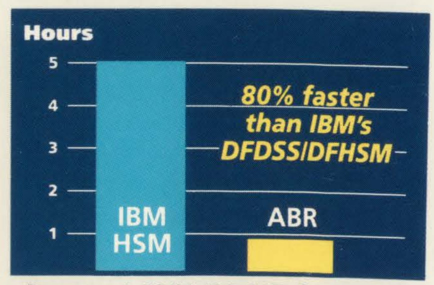
Don't wait until disaster strikes to find your present software is inadequate.

Can Give You That Sinking Feeling

How long will it take to recover your entire data center if disaster strikes?

FDR/ABR recovers DASD volumes from Incremental Backups as if you had done full volume backups each night, in a fraction of the time it takes competing products.

If it takes 5 hours to recover 1 3380-K using IBM's DFHSM, how long will it take for an entire Data center?



Recover 1 3380-K DASD from a full volume and 3 Incremental backups.

Available for IBM MVS, MVS/XA® and MVS/ESA®

Send for a FREE No Obligation 90 Day Trial

INNOVATION® DATA PROCESSING

275 Paterson Avenue, Little Falls, NJ 07424 • (201) 890-7300

CIRCLE #97 on Reader Service Card ▲