ODP-81-666
22 May 1981
21 August 1981

MEMORANDUM FOR: Chairman, Publications Review Board

THROUGH        :   Chief, Systems Programming Division
                   Deputy Director for Processing, ODP
                   Director of Data Processing
                   Deputy Director for Administration

STAT    FROM           :   [                    ]
                   Systems Programming Division, ODP

SUBJECT        :   Request to Participate in a Panel Discussion


1.  I request permission to participate in a panel discussion describing Agency experience in solving a particular type of computing problem.

2.  When approved, I intend to speak at the [          ]Conference in [                    ] during the week of August 23rd. The audience is expected to be comprised of about 400 persons from the United States and Canada representing organizations running similar computer systems.

3.  None of the material to be presented is classified or controversial. I will discuss techniques that we have developed to allow users of IBM's VM/CMS timesharing systems to share data. The architecture of our unclassified Gimini database and AIM electronic mail systems will be described in detail.

4.  I am not under cover and will be identified as an Agency employee. I will also give the standard disclaimer that the views expressed are my own and not necessarily those of the Agency.


/signed/
[                    ]

STAT    ODP/P/SPD/ISB/[                    ]05-22-81/PRBREQ5

Distribution:
    Original - addressee
         1 - VM/SP Project
         5 - Chronos
         2 - DDA

STAT (FROM, left margin)
STAT (para 2, right margin)
STAT (left margin, para 2)
STAT (signature, right margin)

SUBJECT:   Request to Participate in a Panel Discussion

STAT          AUTHOR'S NAME: ☐

TITLE OF PANEL DISCUSSION:   CMS File Sharing


      I have reviewed the outline in paragraph 3 of this request, to the best of my knowledge have found it to be unclassified, and approve it for presentation.


**/s/ Bruce T. Johnson**
_____
Bruce T. Johnson, D/ODP


    2 1 AUG 1981
_____
Date

**/s/ William N. Hart**
_____
Harry E. Fitzwater, DDA


    2 1 AUG 1981
_____
Date

# SHARING DATA IN CMS

The CMS Project of SHARE has submitted a requirement to IBM that begins:

"It is not possible to share CMS files in a multiple-user, read/write environment."

The requirement explains that this deficiency makes it rather difficult to implement applications such as electronic mail and general database systems. We are acutely aware of the truth of this statement.

We have recently implemented two large applications under VM/CMS that share data. Gimini is a refitted version of the GIMS database management system that runs on a S/370 minicomputer. AIM is a facility that provides an automated environment for creating, editing, sending, receiving and filing documents and messages electronically.

GIMS is a hierarchical database management system that runs as a problem program and supports multiple terminals, multiple databases, transaction logging and concurrent transaction processing. It is dictionary driven, allows multiple files within a database, multiple levels of security, and has an EXEC-like capability. In other words, it is a full-function DBMS. We are currently running GIMS under MVS on a ~~3033UP, 370/168~~ and two ~~Amdahl 470/V6s.~~ four 3033 class machines.

GIMS began as a TRW research project back in 1965. The first prototype for an IBM 7094/1410 came out in 1967. Two years later, GIM I was released for DOS/360 and OS/360. GIM II was released in 1972. This is essentially the version that we have been running on our MVS mainframes. In 1979, we decided to refit GIMS to run on a 4331 minicomputer. VM/CMS was the logical choice for an operating system. The first Gimini system went production in April of this year.

Gimini was implemented using a network of virtual machines that boggles the mind. Each database user has an ordinary CMS virtual machine. The software that runs in this virtual submits transactions to the database and displays output on the user's terminal. Each GIMS user is assigned a statement virtual that processes transactions. These virtuals are managed by yet another that we know as the goddess virtual. There is also a tape virtual that allows data to be extracted from, or bulk load to be made to, the database. All communications between the user and the statement or goddess virtuals is performed using the VMCF SENDX protocol. While the statement, goddess and tape virtuals also use VMCF, they make particularly effective use of shared writable discontiguous saved segments.

The data is stored in a small number of large CMS files. Each database may consist of one or more CMS files, each of which corresponds to an "extent." We have a small mod that allows in-place updating of records.

Gimini makes extensive use of discontiguous saved segments. Most of the common code is re-entrant and resides in a 1.3 meg DCSS. There are three shared writable segments containing database system tables. These include the ENQ/DEQ/Wait entries that allow locking for groups of records, compiled copies of the database dictionary and procedures, and lists of

current generation counts for each record. Gimini performs its own lock management, hence it can run in a tightly-coupled, multi-processor environment.

A few functions have been omitted from the initial version of Gimini. There is no history tape; it just isn't needed on a minicomputer. GIMS has a function similar to spooling of consoles; an interface to the VM spooling mechanism needs to be installed. The first version went to single user systems, so multiple databases aren't fully supported. Also, record deadlocks are so rare with only one or two users that the code to handle this condition has been omitted.

Gimini is currently running on ~~two~~ IBM 4331-1s. It took a little over two years for two system programmers to complete the refit. Of the 700 modules in GIMS, about 75 were affected. The changes applied principally to interface routines. Most of the database routines were unchanged, and software development and maintenance continues to be done under MVS. The user's perception of the database is unchanged. Databases are also portable between GIMS and Gimini using tape.

Over the next few months, support will be added for the missing functions. GIM III should be out by June of next year, and it will also run in a VM environment.

AIM provides an enviroment for the creation and management of documents by non-data processing persons. It is similar in structure to IBM's PROFS system, but places greater emphasis on document routing and control. It is also TTY terminal oriented.

The AIM user is provided with more than twenty commands to create and manage documents. Documents live in folders; documents are the logical equivalent of files, while folders are equivalent to minidisks. Get and Put privileges may be assigned to users for folders, while document access is covered by Read, Modify and Append authority. Extensive controls, such as From, To, Through and Carbon Copy, are provided to manage the flow of documents.

The software that runs in the user's virtual machine comprises about 75% of the code. This includes command processors, the database manager interface, a CMS file system interface and SCRIPT subset, all of which run under a tailored version of the editor. The central virtual machine manages the database. It performs the actual manipulation of documents, as well as transaction logging and user notification. Data is transferred between the central virtual and user virtuals via VMCF SENDX.

The data is stored in a structure of CMS files on the central virtual's data disk. Each user has a file that points to a list of folders. A folder has one file that lists all of the documents in that folder, as well as a "shadow" file that lists the users who have access to that folder. A document is comprised of several files: a header that defines routing and symbolic reference data, a text file containing the body of the document, a shadow listing the users who have copies of the document, and annotates or appends containing data added to the file after creation. Each document involves an average of three CMS files. Regardless of the number of users who have copies of a document, only a single copy is retained in the system.

AIM attempts to perform as much processing as possible in the user's virtual to reduce the burden on the central server. There is a powerful interpreter with an alias, or symbolic substitution, capability. All data is double stored, just in case a disk drives goes out to lunch permanently. Most of the code that runs in the user's virtual is re-entrant and resides in a DCSS.

The "pilot project" has been running since January with a select group of 200 users. Next month, AIM will enter "test production" and more users will be permitted to climb aboard. It will enter full production in January 1982, and everybody will be allowed to use the system. We currently have some 2,000 documents in AIM. One projection calls for 300,000 documents when the system is fully operational.

Many additional features are on the drawing board for AIM. It should ultimately interface directly to both XEDIT and Waterloo SCRIPT. It will have to support multiple minidisks and multiple central servers to provide acceptable performance. An archival facility is being developed, as are an array of office automation goodies. It will also be expected to interface to a variety of other applications.

What have we learned from all our efforts? Secure sharing of files requires that the data management portion of an application be removed from the user's virtual machine. Processing cannot usually be concentrated in a single service virtual machine and still provide adequate response. Hence, a careful division of labor between the user's virtual and the central, or intermediary, virtual is necessary.

The performance of the CMS file system tends to crumble when thousands of files are involved. The chief problem arises from the time and storage required to repeatedly search the FST chains. The solution that we, and PROFS, have chosen is to keep our own compressed directory and access only slices of the whole database spread out over multiple minidisks.

We have yet to track any problems down to the performance of VMCF, and it is believed that we could possible improve its performance with a minor mod to the VM Control Program. Finally, shared writable segments have proven to be a very powerful tool. It is, however, necessary to modify the system slightly to prevent unauthorized mucking with the tables.

You can share large amounts of data in CMS. Unfortunately, today's systems require that you spend substantial resources developing the support.

Unclassified

# SHARING DATA IN CMS

Central Intelligence Agency
Washington, DC   20505

STAT

STAT

Foil 1

Problem:

"It is not possible to share CMS files in a multiple-user, read/write environment. This deficiency makes it more difficult than should be necessary to take advantage of the simplicity, efficiency and flexibility of the CMS file system when building very commonly needed interactive applications such as electronic mail, computer conferencing, transaction processing and general database systems.

SHARE CMS Project Requirement 80023

Foil 2

## Applications:

Gimini:       a refitted version of the GIMS database management system that runs on a S/370 compatible minicomputer.

AIM:       a facility that provides an automated environment for creating, editing, sending, receiving, and filing documents and messages electronically.

Foil 3

GIMS:

- A hierarchical database management system that runs as a problem
  program and supports multiple terminals, multiple databases,
  transaction logging and concurrent transaction processing. It is
  dictionary driven, allows multiple files within a database, multiple
  levels of security, and has an EXEC capability.

- Currently running under MVS ~~on 3083JCC, 3701/188 and two 470/16c~~

  ~~four 3033 class mach~~

Foil 4

Chronology:

| | |
|---|---|
| 1965 | TRW research project |
| 1967 | Prototype for 7094/1410 |
| 1969 | GIM I available for DOS and OS/360 |
| 1972 | GIM II available |
| 1979 | Decision to use VM/CMS as a base |
| 1981 | Gimini system in production |

Foil 5

## Components:

- Goddess virtual initializes and shuts down GIMS. Also assigns a statement virtual to a user. Communicates using VMCF SENDX and DCSS.

- Statement virtuals process transactions. There is one for each user signed on to GIMS. Communicates using VMCF SENDX and DCSS.

- User virtuals submit transactions to the statement virtuals and display output on the user's terminal. Communicates using VMCF SENDX. This is the user's ordinary virtual machine.

- Tape virtual allows extracting and bulk loads of a database. Communicates using VMCF SENDX and DCSS.

Foil 6

## Data Organization:

Each database consists of one or more CMS files (extents) in GIMS format. In-place updating of records is allowed.

Foil 7

Discontinguous Saved Segments:

GIMSEG        is a 64K, shared writable segment containing the database "bootstrap," and ENQ/DEQ/Wait tables.  Allows locking at the group (of records) level.

GIMSTATE      is a 1.3 meg, shared R/O segment containing code.

GIMCOMP       is a 3 meg, shared writable segment containing the compiled dictionary and procedures.

GIMSHARE      is a 512K, shared writable segment containing a list of current generation counts for each record.

Foil 8

## Missing Functions:

- History tape.

- Directed output.

- Multiple databases.

- Record deadlock.

Foil 9

## Result:

- Gimini currently running on two 4331s.

- 1.5 system programmers spent 27 months refitting software.

- About 75 out of 700 modules affected. Changes principally applied to interface routines. Database routines are unchanged. Most software development and maintenance is still done under MVS.

- User perception unchanged. Databases are portable between GIMS and Gimini.

Foil 10

Future:

- Provide support for missing functions.

- GIM III available in June 1982.

Foil 11

## AIM:

- Provides an environment suitable for the creation and management of documents by non-data processing persons.

- Similar in structure to PROFS, but greater emphasis given to document routing and control.  Also is TTY oriented.

Foil 12

Externals:

- 26 user commands.

- Documents (logical equivalent of files).

- Folders (logical equivalent of minidisks).

- Get and Put access at folder level; read, modify and append at document level.

- Routing includes from, to, through, carbon copies, plus others.

Foil 13

## Components:

- User virtual software comprises about 75% of code, including command processors, database manager interface, CMS file interface and SCRIPT subset, all running under tailored editor. Communicates using VMCF SENDX.

- Central virtual manages the database. Performs actual manipulation of documents upon request of user virtual, as well as transaction logging and user notification.

Foil 14

Data Structure:

- Folder List.

- Folders and shadows.

- Document headers, text, shadows, annotates and appends.

- All are CMS files (average 3 per document).

Foil 15

## Features:

- Maximize processing in user virtual.

- Interpreter with alias capability.

- Double storing of data.

- Most user virtual code is re-entrant and resides in a DCSS.

Foil 16

## Status:

- "Pilot project" running since January 1981.

- "Test production" in September 1981.

- Full production in January 1982.

- Currently 2,000 documents.  300,000 documents projected at full load.

Foil 17

Future:

- XEDIT and Waterloo SCRIPT.

- Multiple minidisks.

- Multiple central server virtuals.

- Archive capability.

- Office automation goodies.

- Interfaces to other applications.

Foil 18

Lessons Learned:

- Distribute processing.

- Distribute data when there are large numbers of CMS files.

- VMCF innocent until proven guilty.

- Shared writable segments are a very powerful tool.

Foil 18