ABSTRACT

SAS/SHARE software allows multiple SAS users concurrent update access to SAS data libraries and members within the data libraries. The development project for this software was initiated in January of 1985. The product, which enhances the SAS System under MVS and CMS, is currently in alpha test status at several installations.

This paper gives an overview of the historical needs and the conceptual model and then describes in detail the SAS/SHARE software product. Actually, the product appears differently to different SAS users. The SAS programmer sees the product as another means of accessing a data library with new flexibility but with a few new restrictions and concerns as well. The server administrator, who ensures that the product is used properly in the organization, works with the new procedures SERVER and OPERATE provided with the product. The paper concludes with a brief discussion of the implementation details. Thus, the technical complexity of issues increases as the paper develops.

OVERVIEW

Historical Needs

The needs or desires of SAS users to concurrently access data libraries for update has grown significantly in the past five years. Prior to 1981, only PROC EDITOR was capable of updating members of a SAS data library. Still, users were requests by installations with very large data libraries to allow SAS users to share read access to some members of a data library while other users replaced or created different members in the same library. While no one wanted to replace or create the same member at the same time, different members could not be accessed for update at the same time because the data library structure under MVS or the minidisk file system under CMS requires exclusive access to the data library for update with data integrity.

The introduction of PROC FSEDIT to the SAS user community in 1981 expanded the need to concurrently update the same member of a data library. SAS/FSPE software along with SAS/GRAPH software promoted the interactive use of the SAS System. As use of these products has grown, the demand for concurrent update access has grown as well.

Finally, with the introduction of SAS/AF software in 1985, the SAS System greatly enhanced its position as an information center tool. The integrated data entry and data analysis applications often built with the combination of SAS/FSPE and SAS/AF software has mandated development of the SAS/SHARE product. User input from the SUGI software ballot has reflected the need for the SAS/SHARE product. On the 1984 and 1985 ballots, the ability to concurrently update a member of a SAS data library was the third and second highest priority item for the SAS/FSPE product respectively.

Conceptual Model

In the years prior to the development project, several methods of adding concurrent access to the product were considered. In the final development project, the central server model was chosen. Figures 1 and 2 depict the difference in a SAS environment due to this central server.
In the central server model shown in Figure 2, a new SAS execution called the SAS server is added to the environment. This server can also allocate private libraries for read-write access which we call server libraries. The operating system again prevents other users from allocating these libraries to their own sessions at the same time. However, the server is a special SAS execution which can communicate with the other SAS users in the same machine. The server can read and write data to server libraries on each user's behalf. The server is responsible for coordinating multiple user access to the data library so that conflicting user requests do not cause data integrity problems.

**Concurrent Capabilities**

What SAS procedures can cause conflicts? Almost all procedures can cause conflicts in the right circumstances. However, many SAS procedure executions can simultaneously access the same data library and same member as described in order of importance below.

1. When using PROC FSEDIT, two or more users can update the same SAS data set at the same time, but not the same observation in the same data set.
2. If one or more users are using PROC FSEDIT to update a SAS data set, at the same time other users can:
   - copy that SAS data set as well as other members of the library into another library using PROC COPY.
   - run PROC CONTENTS on that SAS data set as well as other members of the library.
   - run PROC DATASETS on the library and browse the data set, but DATASETS cannot modify the data set, for example, with the V selection code. The Contents Menu displays, but the screen cannot be modified. With other members of the library, DATASETS can be used to change member passwords and browse, modify, and delete members. However, DATASETS cannot be used to rename any of the library members.
   - use a SET statement to read the SAS data set as long as the CNTLLEV= data set option is specified.
3. While one user is creating a member in the library, other users can create, read, and update members in the same library.
4. While one user is browsing, copying, and printing a member, other users can simultaneously browse, copy, and print that member.

**USER PERSONALITY**

The primary goal in developing SAS/SHARE software for interfacing with users or SAS programmers is that it be easy to use; that is, to make access through a SAS server almost transparent (as similar to local access as possible). The syntax and semantics of any new statements must be clear and consistent with existing SAS statements. At the same time, advanced programming capabilities, which might conflict with these first goals, must be available for advanced SAS programmers.

At a minimum, it is necessary for a SAS user to be able to specify access to a library through a server. While the SAS System for IBM environments has always required that libraries be allocated with the appropriate IBM system commands, the SAS Systems for mini- and microcomputers use a SAS LIBNAME statement for a similar purpose. In preparation for the SAS/SHARE product, the LIBNAME statement has been added to the Version 5 systems under MVS and CMS in the maintenance release.

**LIBNAME Statement**

The LIBNAME statement associates a libref to a SAS data library. A libref is a logical name used in SAS programs to reference a SAS data library. Prior to Version 5 of SAS software on IBM systems, documentation usually referred to a libref as a DDname. Some users think of a libref as the first name of a two-level SAS data set name.

Besides associating a libref with a SAS data library, the LIBNAME statement under CMS and MVS is used to perform the following functions:

1. allocate a SAS data library to a SAS session
2. specify read-only or read-write access
3. specify server access
4. list libref associations
5. clear libref associations and deallocate a SAS data library.
Under MVS the basic LIBNAME syntax is:

    LIBNAME libref 'dsname' DISP=SHRIOLD
       PASSWORD=ospw SERVER=serverid;

The statement causes the user's SAS execution to dynamically allocate the data library specified by dsname with the access mode specified by the DISP= option as long as the SERVER= option is not specified. If SERVER= is specified, the server identified by serverid is accessed and the library is allocated to the server if necessary.

Under CMS the basic LIBNAME syntax is:

    LIBNAME libref 'userid vaddr rwmode pw'
       SERVER=serverid;

If the SERVER= option is not specified, the user's SAS execution links and accesses the minidisk specified by userid and vaddr with the access mode specified by rwmode. An available virtual address and filemode in the CMS session is chosen by the SAS System. As under MVS, if SERVER= is specified, the server identified by serverid is accessed, and the minidisk is linked and accessed by the server if necessary.

Three other forms of the LIBNAME statement are useful under both MVS and CMS:

1. LIBNAME libref SLIBREF=servlref
       SERVER=serverid; is used to associate a user's libref with a data library that is pre-allocated to the server and has the server libref specified by the SLIBREF= option.

2. LIBNAME libref LIST; is used to list the data library characteristics associated with a given libref.

3. LIBNAME libref CLEAR; is used to free the libref to library association and deallocate the library if it were dynamically allocated by a LIBNAME statement.

The special libref _ALL_ may be specified in the latter two forms to list or clear all librefs. Note that only librefs which have been created with a LIBNAME statement may be listed or cleared. If operating system control language has been used, the LIBNAME statement processor cannot list or clear local user allocations.

Figure 3 depicts the SAS log from an execution in which the user accesses a data library through a server.

    9 libname x 'sas.shared.library' server=share;
       NOTE: LIBREF SUCCESSFULLY ALLOCATED AS FOLLOWS;
       NOTE: LIBREF=X SERVER=SHARE SLIBREF=SYS00546
       NOTE: DSN=SAS.SHARED.LIBRARY
       NOTE: DISP=OLD

    10 proc contents data=x._all_ nods;run;
       NOTE: THE PROCEDURE CONTENTS USED 0.27 SECONDS AND 1144K
            AND PRINTED PAGE 2.

    11 libname y 'sas.readonly.library' disp=shr;
       NOTE: LIBREF SUCCESSFULLY ALLOCATED AS FOLLOWS;
       NOTE: LIBREF=Y
       NOTE: DSN=SAS.READONLY.LIBRARY
       NOTE: DISP=SHR

    12 proc fsedit data=x.usage screen=s.usage;run;
       NOTE: DSN=SAS.SHARED.LIBRARY
       NOTE: DISP=OLD

The first LIBNAME statement specifies access to the data library 'SAS.SHARED.LIBRARY' through the server named SHARE. The LIBNNAME processor accesses the library through the server and prints the library association data on the SAS log. The CONTENTS procedure is then executed on the library accessed through the server. Notice that the CONTENTS procedure statement is the same whether the library is accessed locally or through a server.

The second LIBNAME statement allocates a SAS data library for read-only access to the user's local execution. Compare the SAS log notes issued for the first and second LIBNAME statements. You can distinguish a locally accessed library from a server accessed library because the log notes for the server accessed library include a server name and a server libref.

It is possible for procedures to access members in local and server accessed libraries at the same time. The PROC FSEDIT statement accesses the SAS data set USAGE in the server-accessed library and the screen definition catalog USAGE in the local read-only library. Note that it is more efficient to access a shared read-only library than a server accessed library if all users only need read access. Therefore, this user has stored SAS/FSP screen definitions and data in separate libraries. The data is accessed through the server for shared update, while the screen definitions which are not modified are accessed more efficiently in the read-only library.

Multi-User Example Scenario

In order to show multiple users interacting through a server, another user must be introduced to establish a multi-user scenario. For clarity, we name the user in Figure 3 John. Figure 4 depicts the SAS log from a concurrent SAS session invoked by a user named Mary.

    1 libname y 'sas.shared.library' server=share;
       NOTE: LIBREF SUCCESSFULLY ALLOCATED AS FOLLOWS;
       NOTE: LIBREF=Y SERVER=SHARE SLIBREF=SYS00546
       NOTE: DSN=SAS.SHARED.LIBRARY
       NOTE: DISP=OLD

    2 libname x 'sas.readonly.library' disp=shr;
       NOTE: LIBREF SUCCESSFULLY ALLOCATED AS FOLLOWS;
       NOTE: LIBREF=X
       NOTE: DSN=SAS.READONLY.LIBRARY
       NOTE: DISP=SHR

    3 proc fsedit data=x.usage screen=s.usage;run;
Mary accesses the same libraries in the same manner as John, except that Mary uses the libref X. When Mary enters the PROC FSEDIT statement, observation 1 from the SAS data set USAGE appears on her screen just as it appears on John's screen. However, Mary receives a warning message under the command line which states:

WARNING: THIS OBSERVATION IS LOCKED.
CHANGES WILL NOT BE SAVED.

If we assume that Mary is interested in usage note observations for the SAS/FSP product, she might enter the command:

find prod = 'FSP'

Since observation 41 is the next observation satisfying the command, it appears on her screen. She may make changes to this observation since no other user has accessed this observation. However, if John now enters the same command, he too gets observation 41 but with the warning that the observation is locked.

Suppose John really wants to update that observation. If he wants to know why the observation is locked, he may enter the HELP command. Now the message field displays the following message:

MARY HOLDS CONFLICTING LOCK FOR DATA RECORD IN X.USAGE.

Actually, the TSO userid, CMS userid, or MVS batch JOBname is what appears first in the message. In these examples we assume that John and Mary's userids are their own names.

Assuming John knows Mary's userid, he may call her and request that she move to another observation. If she's out to lunch, then he's out of luck! Let's suppose though that she is editing the observation. She tells John that she'll leave it as soon as she's through. John enters the REREAD command to attempt to obtain the record with update capability. Since Mary has several changes and is still typing them, the server makes John's session wait. After a while, the server gives up and John's session comes back with a message:

SORRY, IT'S STILL LOCKED

Since John really wants to update the record, he enters the REREAD command again and the wait begins again. This time, Mary completes her updates and enters the RFIND command to move to the next SAS/FSP usage note. As soon as the server receives Mary's request for another observation, it completes John's REREAD request. Observation 41 now appears on John's screen with Mary's updates and without the warning message. John can make additional updates to the observation. Meanwhile, Mary's session has continued to request observations from the server until the RFIND command is satisfied.

Server Locking

In order to detect interuser conflicts, the server must record each SAS user's current state for accessed library members and disallow or require to wait any new user requests that conflict with an existing state. This process is referred to as locking.

In the preceding scenario, we observed the use of exclusive control of the record level lock to detect conflicts between Mary's and John's sessions. The first user to access the observation with FSEDIT obtained exclusive control of the record (observation) lock. When the subsequent requests were made by the other user for the lock, it was denied.

What was not apparent in the scenario were other locks held by these user sessions above the record level. Locks above the record level are necessary to protect other users and procedures that might try to access the same member of the same library. The server maintains locks at the library, the member, the data block, and the data record or observation level. The library and member level locks are obtained when the member is opened for data access. The locks do not change unless the member is reopened. The block and record locks are obtained on individual read or write requests. They are released when a different block or record is accessed respectively. By definition, we call the lowest level at which locks are controlled the control level.

There are five different control states for each of these locks: exclusive, shared, update intent, disallow updates, and no control. Exclusive control conflicts with all other control states except no control. Shared control conflicts only with exclusive control. Update intent control and disallow updates control conflict with each other as well as exclusive control. No control does not conflict with any other state.

The SAS System requests these control states at the various lock levels to maximize concurrent access to data libraries through ensuring data integrity. Write accessers (update and output open modes) always request exclusive control of the lock at the control level and update intent control of all higher level locks. Read accessers (input open mode) obtain disallow update control at the control level and shared control at levels above. By default, no control is obtained for read access below the member level.

By default, the SAS System uses a control level of member to ensure data integrity. Certain SAS procedures override this default. In particular, PROC FSEDIT has a default control level of record. PROC CONTENTS and PROC COPY use control levels of block. PROC DATASETS uses different control levels for different functions. Most PROC DATASETS functions obtain exclusive
control at the member level. The full-screen browse function obtains shared control at the member level while the rename function requires exclusive control at the library level.

If we return to the John and Mary scenario, we can observe how some of these other locks work. When we left John and Mary, both were executing PROC FSEDIT with a SAS data set of usage notes accessed through the server. Let's assume that Mary continues her PROC FSEDIT session while John terminates PROC FSEDIT to do some data analysis.

In particular, assume John wants a sorted report of usage notes from the USAGE data set. Figure 5 shows John's SAS log if he simply tries to sort and print the SAS data set in the server accessed library.

Since PROC SORT tries to replace the SAS data set (that is, it opens the data set for output), the exclusive control request at the member level fails, and an appropriate error message is written on the SAS log. Note also that the PROC PRINT step fails in a similar manner. It uses the default input control level of member with Mary's lock. John needs some advance programming features to overcome this problem.

Advanced Programming

As John discovered in the scenario above, advanced SAS programmers may want to override the default control levels where they know no harm will occur. By lowering the control level, you can change the requested lock state at higher levels to cause fewer conflicts. You can use the CNTLLEV= SAS data set option to override the default control level in some cases.

The CNTLLEV= option specifies a user requested control level for access to a SAS data set. The user requested control level will override the default control level where allowed and where it doesn't obviously cause data integrity problems. CNTLLEV= REC, CNTLLEV=BLK, and CNTLLEV=MEM are the appropriate specifications for record, block, and member control level respectively.

In Figure 8, John's SAS log shows that he has specified CNTLLEV=BLK in the SET statement in his subsetting DATA step.

Since PROC COPY uses a control level of block, it can run concurrently with PROC FSEDIT against the same member, but it creates a copy of the data that does not reflect any updates a user is currently making to an observation with PROC FSEDIT. Note that John cannot use PROC SORT DATA=WORK.USAGE instead of PROC COPY for the same reason that PROC PRINT cannot be used above.
The CNTLLEV= option allows concurrent access of the member while Mary is still editing it. Like PROC COPY, the DATA step does not obtain any updates which Mary is making on her screen.

Finally, Figure 9 depicts the SAS log when John tries to use the CNTLLEV=BLK option in the DATA statement to override the exclusive member level control for output opens.

```
36 data x.usage(cntllev=blk);
   ERROR: MARY HOLDS CONFLICTING LOCK FOR X.USAGE.
37 set x.usage(cntllev=blk);
38 if prod='FSC'
39 then prod='FSP';
40 run;
```

NOTE: SAS STOPPED PROCESSING THIS STEP BECAUSE OF ERRORS.
NOTE: THIS DATA STATEMENT TOOK 0.15 SECONDS AND 1148K.

Since a data set opened for output requires exclusive member level control, the CNTLLEV= option is ignored and the output open still conflicts with Mary's update intent lock at the member level.

SERVER ADMINISTRATION

Server Administration is a new human function required by the SAS/SHARE product to execute, control, and operate one or more SAS servers and to inform the SAS user community of installation rules and requirements for server use. The basic administration functions are server operations, serverid administration, and server access control.

While SAS/SHARE software has been designed so it can be used with a minimum degree of administration, some installations may choose to tightly control and administer use of the software. Just as installations will differ greatly on the degree of administration, so will they differ on the way they assign the administration functions to people within the user organization. Therefore, server administrator refers to whoever is responsible for the functions.

Server Operations

Server operations is the function of server creation, server execution control, and server termination. In many installations the server administrator will work with the data center operations personnel to determine the best way to perform these functions and may turn over the functions to the data center personnel.

To create a SAS server, the server administrator need only execute PROC SERVER from any SAS System execution where SAS/SHARE software is installed. This means that a server can run under MVS as a TSO session, a batch job, or a started task. Under CMS, a server can execute in a CMS session (interactive or disconnected), a CMS batch machine, or a service virtual machine. The batch environments should be avoided under both MVS and CMS.

While the server procedure can run under any SAS environment, it is not necessary to have a full-function SAS environment for the server's execution. It is recommended that you run the server in a non-DMS SAS environment execution. The server requires access to the WORK data library and the SAS log (FT11F001). Other files and libraries other than those libraries which are pre-allocated for shared access are not required (for example, the standard SAS print file (FT12F001) is not used by the server).

The standard PROC SERVER syntax is:

PROC SERVER options;

where options is one or more of the following:

SERVER=
SERVERID=
ID=
PASSWORD=
PW=
TRACE
TRC
MSG

specifies the eight-character serverid or name by which the server is identified under MVS. Under CMS, this option is ignored and the CMS userid is used. If the option is not specified under MVS, the TSO userid, JOBname, or STCname of the SAS execution is used.

specifies the installation password, if any, required to run PROC SERVER. When SAS/SHARE software is installed, a password may be established to limit execution of PROC SERVER to people who know the password. If such a password is established during installation, this option is required to run PROC SERVER.

specifies that the server is to be started with the trace facility active for default trace types. If this option is specified, a variable blocked sequential file must be allocated with a fileref of SRVTRACE to which the server writes trace records. A trace may be required by SAS Institute to help determine system problems should they occur. However, running a trace may affect the performance of the server.

specifies under MVS that the server will display a message on the system console during termination (either normal or abnormal). The message is highlighted and nonrollable to ensure that an operator is notified when the server terminates.
Figure 10 depicts the SAS log from an example server execution.

```
1 proc Server server_id='SHARE' pW='XX' trace; run;
```

TRACE 000001 SERVER SHARE STARTED
TRACE 000003 SERVER LIBRARY SYS0001 ACCESSED AS LIBRARY X BY USER JOHN
TRACE 000006 X USAGE OPENED FOR INPUT BY USER JOHN
TRACE 000010 SERVER LIBRARY SYS0001 ACCESSED AS LIBRARY Y BY USER JOHN
TRACE 002264 X USAGE REOPENED FOR UPDATE BY USER JOHN
TRACE 002308 X USAGE (OPENED FOR UPDATE) CLOSED BY USER JOHN
ERROR: TRACE DATASET OUT OF SPACE - WILL REUSE
TRACE 000015 Y USAGE OPENED FOR INPUT BY USER MARY

In this example, a server with the serverid of SHARE is created. Since the installation has restricted the use of the SERVER procedure with a password, the PW= option is supplied but the value is overwritten with X's on the SAS log. The TRACE option starts the trace to the file referenced by SRVTRACE.

When the server is initiated, it prints the first log message, which states that it is started, and then waits for user requests. As user requests are processed, other log messages are printed for key events. In the example, you see messages indicating when libraries are accessed with LIBNAME statements, when members are opened and closed, and when libraries are released through LIBNAME statements or SAS user termination.

Since the TRACE option is in effect, the log messages are preceded with the word TRACE and the number of the corresponding record on the trace data set. This can help isolate records of interest in the trace file. Note in the example that the trace data set ran out of space. An error message is written to the log and the trace wraps. This overwrites earlier trace data. Therefore you should have a large trace data set when the file is on DASD or use a tape file which will not run out of space.

Once a server is executing, it may be necessary to control its execution. The server resources that one may wish to control include users, libraries, members, the trace, and function or request queues. Control actions that one may want to perform on these resources include display, stop, start, and quiesce. Most of these resources will be controlled through the PROC OPERATE procedure provided with SAS/SHARE software. However, in the alpha test version in the field, OPERATE can only be used to terminate a server. The ability to control libraries, users, and the trace must be developed prior to production. Future releases will add the missing functions as users require the functions. The exact syntax of statements discussed below and shown in the examples may change during implementation.

The OPERATE procedure is a line-mode procedure that can be used to control one or more server executions. It can be executed in any SAS environment just as PROC SERVER can. Execution of PROC OPERATE in a batch environment is useful for executing "canned" operator commands such as starting and stopping libraries at certain times or terminating the system.

The standard PROC OPERATE statement syntax is:

```
PROC OPERATE options;
```

where options is one or more of:

- **SERVER=** the eight-character serverid of the server to which the first commands apply. If this option is omitted, a SET SERVER= command must be used prior to any other commands.

- **PASSWORD=** specifies the installation password, if any, required to run PROC OPERATE. This is the same password required to run PROC SERVER.

- **PW=** specifies the installation password, if any, required to run PROC OPERATE. This is the same password required to run PROC SERVER.

Following the PROC statement, one or more OPERATE commands can be entered prior to the RUN statement. If PROC OPERATE is being executed interactively, the commands can be entered one at a time and are executed immediately.

The planned command statement syntax for PROC OPERATE is:

```
verb objecttype = object | (objectlist);
```

where

- **verb** is STOP, START, QUIESCE, DISPLAY, or SET

- **objecttype** is USER, LIBRARY, TRACE, SYSTEM, or SERVER

- **object or objectlist** is the name or names of the objects to be processed.

The STOP command immediately terminates processing against the named resources. Any active user work for the stopped resource will receive an error when subsequent attempts to use the resource occur. This may cause SAS user execution abends.

The QUIESCE command is a more graceful means of terminating processing against resources. It prevents new work from beginning but allows existing work to continue.

The START command reverses the stopped or quiesced status of the named resources.

Use the DISPLAY command to obtain information about the named resources. The information is written to the SAS log. It appears immediately on the SAS log for interactive executions.
The SET command is only used to change the name of the server to which subsequent commands are sent. The syntax is:

SET SERVER=serverid;

The SET command does not validate the server name. The validation occurs when the next command is sent to the named server.

Figure 11 depicts the SAS log from a PROC OPERATE execution.

```sas
1 proc operate server=share pw=xxxxxxxx;
2 display user=_all_;

USER STATUS LIBRARIES
JOHN ACTIVE X SYSDATA
MARY ACTIVE Y SYSDATA

3 stop user=mary;
4 display user=mary;
5 quiesce user=john;
6 display user=john;
7 start trace=(function,conflict);
8 start user=sry;
9 stop system;
10 run;
```

By default, the commands are sent to the server SHARE for execution. The password is supplied on the PROC OPERATE statement because the installation has restricted SAS/SHARE software procedures.

The DISPLAY statement asks for basic information on all users of the SHARE server. The users are listed with their status and a list of libraries which they are accessing. The operator decides to stop Mary's execution immediately with the STOP command (maybe because she is out to lunch without releasing a data set).

Note: if Mary is editing an observation with PROC FSEDIT, she will lose the updates on her screen. Prior updates are not lost. After stopping Mary, the operator displays her status to ensure that the command is processed. Note that no libraries are listed for Mary as they were released by the STOP command.

The operator decides to QUIESCE John in order to allow him to finish the work he is doing. In the subsequent DISPLAY response, John is still accessing library X. When he closes all members open in that library, it will be released. When all libraries are released, he will be set in the stopped status.

The START TRACE command starts the trace facility if it is not currently active and adds the trace record types associated with function processing and conflict detection to the list of trace records cut. For the trace to work, the trace data set must be allocated with a fileref of SRVTTRACE. A corresponding STOP trace command can be used to remove record types from the list of trace records to cut. If all records are removed, the trace is stopped.

When the operator wants to allow Mary to access the server once more, he issues the START USER command. This does not establish a communication path between the server and Mary. It simply allows Mary to re-establish a path with a LIBNAME statement.

To terminate a server, the STOP SYSTEM command is issued. Normally the operator should quiesce the system first and try to allow users to complete their work. Finally, the RUN statement is used to terminate the line-mode execution of PROC OPERATE.

Serverid Administration

In addition to server operations, the server administrator must at a minimum establish name(s) for the server(s) and inform the user community of the name(s) to use. The names of the servers in the installation must be unique, as that is the manner in which a user identifies which server to access. Increasing the number of servers at an installation increases the importance and the burden of server administration.

Why might installations choose to use multiple servers? There are many reasons, foremost of which are performance and application importance. The server, as implemented in this first release, is a single-threaded system. This means that it is not possible to concurrently process two users' request within one server even on a multiprocessor machine. In the current implementation, all input/output operations are effectively synchronous. Future releases may be able to address this limitation. Having multiple servers for separate applications, departments, or some other division of the user community spreads the workload across the servers and reduces the degree to which any one server bottlenecks multiple users.

Just as important as performance, may be the desire to isolate sensitive or important company applications from other SAS users in the organization. This again may be done to enhance performance but may be done for data security as well. In order to access a data library, the server must be authorized by any installation security package to do so. One of the many ways to secure the data is to restrict access to servers to those people who the company authorizes.
Server access control

Many installations will choose not to control access to SAS servers. Of those who do want to control access, many simply want to prevent a casual user from creating a server or using PROC OPERATE. Finally, some who do choose to control access have many unique requirements for a control system in their environment. Therefore, in order to speed delivery of this product to the user community, the initial product does not try to solve everybody's requirements.

The installation product password is sufficient to meet the needs of the first two groups of installations. In addition, normal SAS data set protection with the READ= and PROTECT= data set passwords is still enforced when accessing SAS data sets through a server.

Some applications may require that a data library always be accessed through a server when the server is active. To ensure that no SAS user allocates the data library to his own execution, the data library can be pre-allocated to the server prior to executing PROC server. This pre-allocation must be done with TSO commands, JCL, or CMS commands to prevent the library from being de-allocated after its first use.

For installations which require more sophisticated access control, user exits are provided from the SAS code. Installations can write assembler code for these exits to interface with the installation security package and tailor the controls. For example, an installation may wish to use the installation security package to limit access to particular servers by userid or to particular data libraries by userid. Sample RACF user exits for these functions are provided with SAS/SHARE software. If an installation with ACF2, TOP SECRET or other vendor-supplied security packages wants to contribute sample exits for these products, they will be made available as well.

IMPLEMENTATION AND FUTURES

Detailed discussions of the IBM operating system facilities used to implement communications between multiple SAS executions is beyond the scope of this paper. This section simply names the facilities and states plans for future implementations.

Under MVS, the cross memory services functions are used to pass data between the server's address space and users' address spaces. The SAS SVC0 routine is used to obtain authorization to establish the cross memory services environment.

Under CMS, the VM InterUser Communications Vehicle (IUCV) is used to pass data between the server's virtual machine and the users' virtual machines. The server's virtual machine directory entry defines the maximum number of users with which the server can communicate concurrently.

Under development at the Institute is an implementation which uses VTAM within an MVS multi-CPU environment to allow a server to execute on one CPU and users to execute on another CPU within the VTAM network. We are testing this code at the current time.

We are investigating the feasibility of communicating between VM and MVS systems via the new VTAM support in VM. This would allow users of a MVS or CMS system to access libraries through a server under the other operating system. This is a long range objective if feasible.