Experiences with Remote Library Services: Choosing the Right Platform


ABSTRACT

The Management Information Systems (MIS) department at SAS Institute Inc. uses SAS® software to develop all in-house applications. Many of these applications use remote library services (SAS/SHARE®) to allow multiple-user access from multiple platforms to the data simultaneously. This paper discusses the following MIS experiences in using remote library services:

- Deciding what operating system to house the data when users from various operating systems will be accessing it.
- Fine tuning for performance.
- How to begin with remote library services and the differences on various platforms.
- Auto-backup issues.
- SAS/SHARE server maintenance.

The Problem

As the technology available to a SAS applications developer increases, it is easy to lose track of the seemingly endless ways to program various tasks. In addition, our client/server world makes knowledge of multiple operating systems imperative. It seems there must be a way to simplify the simultaneous access of data from multiple platforms. Fortunately, there is an easier way... SAS/SHARE software.

Like everyone else, MIS at SAS Institute has experienced these problems too. As applications are developed, they are required to interact with each other more and more. The data has to be accessible from every platform, and users must be able to access this data from anywhere, in a painless way.

A Solution

More and more companies are running several different operating systems. Users throughout the company need to access their applications regardless of what platform they are based on. SAS/SHARE provides an easy way of doing this. With a SAS/SHARE server in place, these things are taken care of for you. Multiple users can simultaneously access data on various platforms using standard TCP/IP.

The experiences described in this paper are primarily for UNIX, MVS, VM, and Windows 95 although the example will be beneficial to all operating systems. This paper provides useful information and concepts to help you design and set up your multi-access and multi-user SAS applications.

What Operating System to Use

The first step is deciding on what operating system to house the data. The following issues are crucial to making this decision:

1) Are the users limited to a particular operating system by hardware constraints?
2) Is one operating system more saturated than the others?
3) Is security an issue?
4) Do the users have a strong opinion about the platform on which the application should reside?
5) Which operating system do most of the users access daily and feel most comfortable with?
6) If you want a GUI interface, which operating system will provide the best look and feel?
7) On which operating system will the application get the best performance?
8) Which operating system has the needed resources?
9) Is the new application replacing any existing application(s)?
10) Does the application need to interface with other applications? If so, on which operating system do they exist?

Answers to these questions should provide you with enough information to make the best decisions for your computing situation. Any constraints that exist to limit you to a particular platform should begin to surface.

In our deliberations on where to place the data for a new asset/network management application, we used these questions to evaluate our own needs. The platforms from which we had to choose were: MVS, VM, UNIX, and Windows 95. The following information was gathered to answer the questions listed above.

1) Are the users limited to a particular operating system?

Your users could be limited to a particular platform because of hardware constraints, cost effectiveness, or time and so on. When we began development of the new application known as INSINC (Institute System for Inventory and Network Consolidation), ISD personnel gathered information and found that few users were limited to only mainframe access because of hardware constraints. If we wanted to house INSINC on a non-mainframe platform, management was willing to provide the hardware for the few users that needed to convert.

One year prior to analyzing this situation for the development of INSINC, we performed the same analysis for another new application. At that time, there were many users that could only access MVS or VM because of hardware constraints. It would have been very costly to buy new hardware for this large number of users, so the decision back then was to go with MVS for that application. Time has a way of changing things.

2) Is one operating system more overloaded than the others?

If you have a choice of operating systems you will want to take into consideration the present and projected workload versus capacity on each platform. In our case, MVS and VM CPU was in great demand by many divisions within the company. In priority, we fell a bit lower than our fellow divisions. The capacity of the mainframe could not accommodate everything. Making the decision NOT to use one of these platforms was fairly straight forward for this reason in particular. The performance being experienced on MVS and VM at the time was less than desirable for an interactive application.

3) Is security an issue?

Each operating system being considered has its own set of strengths and weaknesses. Both MVS and VM have strong security software available. RACF and VM/Secure are both very effective. UNIX and Windows 95 do not have the same strength for security purposes but they do offer other strengths. The application being developed at the time did NOT need tight security restrictions although it did need some. The security that exists on UNIX along with some passwords on SAS datasets was sufficient for our purpose.

4) Do the users have a strong opinion about the platform on which the application should reside?

We considered user preference to be an important factor in our decision. Strong likes and dislikes can be a major factor in the success or failure of an application. Users generally prefer the operating system they are most familiar with and the one they use most to do their work. Your application will be better received and more likely to be used if it is easily accessible. We are also service oriented and try diligently to please our users whenever technicalities will allow.

5) Which operating system do most of the users access daily and feel most comfortable with?

This goes hand-in-hand with #4. Users are most likely to desire an operating system that they feel most comfortable with. If they are on
Systems Architecture

a particular system to take care of daily work, they will want the applications they use to be on the same system. The answer to this question may very well be more than one platform. With cross-domain services, you can accommodate them simultaneously. You still need to decide where the majority of your users will be accessing the application. It may be that the data should be housed on the most 'popular' operating system.

6) If you want a GUI interface, which operating system will provide the best look and feel?

Not all operating systems have the same look and feel to their GUI interfaces. Some features of SAS/AF® widgets will look a bit different on one platform than they do on others. The best GUI look and feel is on the open systems platforms (i.e., UNIX and Windows 95). While the mainframes still look good, they don’t have quite the same GUI appearance.

7) On which operating system will the application get the best performance?

When answering this question, also consider the information gathered in answering #2. The workload of the operating system will definitely have an impact on the performance of your application. When users are struggling through your application and are not pleased with the performance, they will call you. They will not realize that the operating system is the primary factor in this situation. You cannot predict the future but you can rationally analyze the computing resources of your company today and try to remain flexible enough to move the data in the event you need to. Using cross-domain services will easily allow this to happen.

8) Which operating system has the needed resources?

There are several issues to consider when answering this question. Resources such as storage space, job schedulers, security software, network protocol, and electronic mailers are just a few examples of system items that you may desire to use within your application. For example, if you wanted to send electronic mail from an SCL method, you would need to have a system mail command that would pass the mail file to the mailer and actually send the electronic mail. If you needed to run some background jobs in the evening, your platform would need to have some kind of job scheduler (Control-M, cron, and so forth).

9) Is the new application replacing any existing application(s)?

If a new application is replacing one or more existing applications, you must guarantee that everyone using the existing systems can still access the new system. If two older systems existed on MVS and were replaced by a new application on UNIX, you would have to ensure that all users of the old systems had adequate hardware and training to access the new system. When designing INS INC, we knew this was the case. We were replacing one old system that existed on MVS and another old system that existed on UNIX. We researched and found that all users did indeed have hardware and knowledge sufficient to access the application on UNIX.

10) Does the application need to interface with other applications? If so, on which operating system(s) do they reside?

If the application must interface with an existing application, then that application and/or its data must be accessible. In our example, INS INC definitely had to interface with an existing application both in interface and data. That application existed both on UNIX and MVS therefore INS INC must also exist on these two operating systems. In this situation, only the data needed to be accessible from MVS and UNIX while the interface itself need only be accessible from UNIX.

MIS Analysis and Conclusion

Our conclusion was to place the data and the SAS/SHARE server on UNIX along with the application. By using the cross-domain server, we kept the ability to access the data and run background reports from MVS as well. The applications on MVS needed only change the name of the server they used to access the data.
Even though we made the right decision at the time based on all the facts, our user base has since then migrated and so must the application. We have found that more and more of our user base has converted from UNIX workstations to PCs and are using an X emulator to access UNIX and invoke the INSINC application. However, the data does not necessarily have to move at all. Since it is already set up for cross-domain access, we will most likely leave it on UNIX. Only the catalogs will have to be copied to the PC. This will allow our application to run native on both UNIX and the Windows 95 but they will both access/update data on UNIX.

In light of that, we plan to copy our catalogs to a Novell server on the PC network and to use cross-domain access to the data and SAS/SHARE server on UNIX. Or, perhaps we will install the INSINC catalogs directly onto each PC and use cross-domain access to the data on UNIX. We will have to analyze the situation before doing so. The PC network is new to us and so it will be a new experience to figure out how best to approach production application installations.

As you can see, this is an ongoing iterative process. You need to continually ask these questions to keep up with your users, technology, and the competition. This also reinforces the importance of writing applications so they are portable. We have done so with INSINC so the move of our catalogs to the PC will be straightforward.

**Fine Tuning for Performance**

We found that several things could be done to directly impact the performance of the INSINC application. Since we opted for UNIX, we had a choice of file services to use in handling the data. Through much experimentation and consultation with our UNIX support team, we found that the data should exist on the local disk of the actual workstation that is running the SAS/SHARE server. This removes any networking that needs to be done between the server and the data. We also eliminated another layer by not using file services like afs or nfs.

Performance on MVS and VM can be greatly improved by having the data and catalogs on cached DASD volumes. We found this to be very beneficial. You will also want to consider the other data that is housed on the same DASD HDA. If you are contending with a paging volume on the backside of the HDA, you will most likely experience a considerable performance degradation. Our MVS and VM support personnel were very effective in pooling data to the proper DASD volumes and made sure highly used data was not in contention.

Performance on UNIX and a Windows 95 can be improved by increasing the memory. Depending on the number of users and amount of workload on your particular hardware, you may need a memory increase. That will also need to be monitored and evaluated as an ongoing process during the life of your application. You will also want to monitor the workload on your hardware from other sources. There is only a finite amount of CPU to go around and you have to be sure your application can get what it needs.

**How to Begin with Remote Library Services**

A SAS/SHARE server runs in a separate SAS session that serves users' SAS sessions by controlling and executing input and output requests to one or more SAS data libraries.

When you allocate a SAS data library through a SAS/SHARE server, your SAS session reads from and writes to that SAS data library through the server instead of reading and writing directly to the library. A SAS/SHARE server is established in a separate SAS session invoked by a server administrator. Further detail can be found in *SAS/SHARE Software, Usage and Reference, Version 6, First Edition*.

The following steps will help you in getting started with your SAS/SHARE server and allow it to use remote library services.

The examples given are for creating a SAS/SHARE server called INSINC which will be running on a UNIX workstation with a node name of MIS01. Simply substitute all...
occurrences of \textit{INSINC} with the name of the server you are trying to start. Substitute all occurrences of \textit{MIS01} with the node name of the UNIX workstation on which you are trying to run the server. Steps 1 through 3 below cover the required activities. Further steps are optional but listed here for your knowledge and future reference.

1) Defining a TCP port

A TCP port is an end point for communication between applications or devices and generally referred to as a logical connection and provides queues for sending and receiving data. Each port has a number for identification. This number (TCP port number) identifies a particular TCP resource within a TCP/IP node.

A TCP port number must be defined for the server. Choose a name for the server, define a TCP port number for it, then make sure this entry is added to the etc.services on the UNIX workstation. It will also need to be defined in the etc.services file on any PCs that want to communicate with it. You may need to have a systems programmer at your site do this for you.

\textbf{Example line from etc.services:}

\begin{verbatim}
insinc 5555/tcp # INSINC server
\end{verbatim}

2) SAS source code to start the SAS/Share\textsuperscript{6} server

The following example SAS source code is used to start the SAS/SHARE cross-domain server. It can be run from the SAS PROGRAM EDITOR of an interactive SAS session that has been invoked on the UNIX workstation where you want the server to run, or it can be invoked via a script executed on the UNIX workstation where you want the server to run. (example of script given later).

For testing, an interactive SAS session is fine. For production, a script is recommended. After submitting this through the PROGRAM EDITOR, the SAS session remains busy and cannot be used for anything else because PROC SERVER remains active (the server is running). This also makes it easier for others (such as data center operators) to start and stop the server.

\textbf{Start server SAS source code:}

\begin{verbatim}
options comamid=tcp;
proc server server=insinc log=(all,query);
run;
\end{verbatim}

By using the \texttt{options comamid=tcp;} statement, the server becomes available for TCP/IP access. This means the server has been defined as a cross-domain server. Users can access the data allocated through this server from any operating system at your site, provided that operating system uses TCP/IP. If \texttt{options comamid=appc;} had been used, the server would be accessible using APPC from various platforms.

The \texttt{comamid} option specifies the access method that can be used to interact with this server besides XMS which is the default.

The \texttt{log=(all,query)} specifies that the server report all types of server resources and SQL queries. This information would be included in the SAS/SHARE server log.

3) Script to start the SAS/Share server

The following is a sample ksh script used on UNIX to start up a SAS/SHARE server for INSINC. The script as shown below must be run on the actual UNIX workstation that you want to run the SAS/SHARE server. For the example being demonstrated, \textit{INSINC} is the name of the SAS/SHARE server and \textit{MIS01} is the node name of the UNIX workstation that you want the server to run on.

\textbf{Source for ksh script:}

\begin{verbatim}
SAS start_server_insinc.sas । । -sasuser /u/insinc/sasuser । । -work /u/insinc/saswork । । -unbuflog \ । । -filelocks continue \ । । -log /u/insinc/insinc_server_log । । -memsize 84m
\end{verbatim}

All of the options specified on the SAS invocation above are optional but quite helpful. More information and options can be found in the SAS Companion for the UNIX Environment.
and Derivatives and also in the SAS Language, Reference, Version 6, First Edition. An explanation of each follows. Information can also be found in any of the companion manuals for whatever operating system you are interested in.

SAS start_server_insinc.sas - This executes SAS and passes in the source that is in a file called start_server_insinc.sas. This file contains SAS source code like the example given in #2 of this section.

sasuser - This option specifies that you want a particular file used as the SAS user profile rather than using the default.

work - This option specifies that you want a particular file used as the SAS WORK library rather than using the default.

unbuflog - This option specifies that you want lines written to the server log as they occur, instead of buffered up and written to the log at the time the server ends. This is especially helpful because you can read through the server log as transactions occur rather than waiting until the server is stopped.

filelocks - This option specifies how we want SAS to control file locking. filelocks continue specifies that file locking is in effect. If a file cannot be locked, the file is opened and a warning message is sent to the log. In order for file locking to work on some machines, you must have the machine's "file locking service" running. Refer to the SAS Companion for the UNIX Environment and Derivatives for more information.

log - This option specifies that we want the server log written to a particular file rather than just using the default. We like to put a time/date stamp on the name of this file from within our ksh script. This way, each server log is clearly marked as to the time frame it refers to. The time/date stamp naming convention is not shown in the example given here. We keep the most recent 7 server logs online. This way we can go through a week of server logs in the event we need to investigate an occurrence.

memsize - This option specifies the limit on the total amount of memory to be used by this SAS session. Enlarging this can greatly improve the performance of your server.

4) Stopping the SAS/SHARE server

The server can be stopped in much the same way that it is started. A SAS session or UNIX script can be run to have the server terminated.

Stop server SAS source code:

```sas
options comamid=tcp;
proc server server=mis01.insinc;
stop server;
run;
```

The server can also be stopped by killing the process id. This is the method we use most often.

5) Using the server now that you have it

Once you have the SAS/SHARE server and the data, you simply allocate the data using LIBNAME statements. The only difference here in going cross domain (2nd example below), is the need for the proper comamid option and fully qualified host name.

If you access the server on the same platform, there is no need to set the comamid option. You would want the default access method of XMS to be used as in the following example. Our examples given here use TCP/IP

LIBNAME allocation to the insinc server from UNIX:

```sas
libname insinc '/ulinsinc/data' 
server=mis01.insinc;
```

If you access the server cross domain (from another operating system), you need to specify the access method as TCP or APPC and provide a fully qualified host name as in the following example.

LIBNAME allocation to the UNIX insinc server from MVS:

```sas
options comamid=tcp;
%let mis01 = mis01.unx.sas.com;
libname insinc '/ulinsinc/data' 
server=mis01.insinc;
```
6) Other issues

You can run a server in a secured mode if you want to. If a secured mode is desired, the server will query users for their userid and password. This will validate them through the security system that is applicable to your operating system that houses the data and the server. For example, let's use MVS with RACF. If a secured server was running on MVS and a user wanted to allocate some MVS data to a UNIX SAS session, the user would be queried for their userid/password and would then be RACF validated. The userid is not actually logged onto the system. To have a server run in secured mode, simply set the TCPSEC macro variable.

Start secure server SAS source code:

```plaintext
options comamid=tcp;
%let tcpsec= _secure_; proc server servef=insinc IO!J=(all,query); run;
```

LIBNAME allocation to a secure UNIX server from MVS:

```plaintext
options comamid=tcp;
%let tcpsec = _prompt_; %let mis01 = mis01.unx.sas.com;
libname sinc '/ulinsinc/data' server=mis01.insinc;
```

The TCPSEC can also be set to userid.password instead of _prompt_. This sets the TCPSEC to whatever you want instead of prompting the user for it. We have situations where we store userids and passwords in an encrypted form and retrieve them and set the TCPSEC for the user. This way the user is not annoyed with a query every time they bring up that particular application which accesses a secured server.


Auto-Backup Issues

If you have a SAS/SHARE server managing your data you will need to be aware of some auto-back issues. Assuming you wish to create a backup of your data each evening, you will want to ensure that you attain that backup without fail. It will be necessary for you to investigate the way the backups are set up on your particular operating system. For example some packages take incremental backups only if the data is not in use. Having the data allocated to a SAS/SHARE server automatically means it is in use. Therefore, the incremental backup would not be attained.

What we have done to get around this situation is to stop the data on the server each night, create our backups, then put the data back on the server. This is done at a specified downtime each night so users know when the application will not be available on a regular basis. We also take this opportunity to start a new server. This keeps our server logs more manageable. Each server log is for a 24-hour period so we can easily choose the appropriate log when trying to trouble shoot. This also gives you the chance to process your data. Resorting and Reindexing your data for example, cannot be done while the data is being used. You must have exclusive access to perform these types of data manipulations.

SAS/SHARE Server Maintenance

We do regular preventive maintenance. The following are simply precautionary steps but they will also benefit you in the event a problem does happen to occur. They are simple steps that will aid you in maintaining your SAS/SHARE server.

Keep server logs

We recommend you keep server logs for the last 7 days. These are very useful in tracking down things that may have 'happened' to your data. The server log will identify users and show what they did or attempted to do to your data. We've been able to find problems and anomalies easily by searching through the server logs. The logs are also beneficial in debugging applications. Error or warning messages will be issued to the log that will aid in the trouble-shooting process.
Restart the server regularly

We like to restart our SAS/SHARE servers each night at a predesignated time. This way, each server log is for a 24-hour period. This greatly simplifies trying to look through the logs.

On UNIX we are able to 'own' our servers and do with them as we will. On MVS, the systems programmers 'own' the servers and we have no say in how often they are restarted or how the logs are kept. Typically, the servers are only restarted whenever an IPL is performed. As a general rule, this occurs once each week for us. This has no impact on the performance or state of the server. It merely gives us much larger logs to have to look through when trying to troubleshoot a problem.

Auto-monitor the server

On UNIX we have a ksh script which monitors the server to make sure that it is up and active. The monitor uses a perl script called pingshare to see if the server is responding. Pingshare is a perl script written by Paul Kent at SAS Institute Inc., that works only on UNIX. (Send electronic mail to kent@unxsas.com to receive more information.)

We use the monitor to restart the servers each night and to restart them if they should be stopped by some other means during the day. The monitor sends notification to the appropriate support personnel and restarts the server. This way, the server is restored the fastest way possible. I usually receive notification that it has been restarted before I even realize that it had been down. Once this notification has been received, it triggers support personnel to start looking for a problem that has caused the server to go down. For example, an afs volume server may be down, some networking protocol may be having trouble, and so on.

The server monitor has proven to be a very useful tool but again, this is optional and not required. Following, is an example of the server query portion of the server monitor.

Server monitor ksh script:

```bash
#*** Are the servers up and responding? ***
pingshare $server | grep -c passed | read rc

#*** Crank up server if it's down ***
if ((( ! $rc )); then
    start_servers ${server}
    zwrite $notify -m ${server} server
    restarted on $(hostname)
fi

$($server), $notify, and $(hostname) have been set previously in the ksh script. $($server) is the server name, $notify is a list of support personnel userids, and $(hostname) is the name of the UNIX workstation that the server and monitor run on. For our example, $($server) would be insinc, $notify would be teresia, and $(hostname) would be MIS01.

The zwrite is a UNIX function that pops up a message window to the recipient containing the specified text. A mail command could just as easily be used to send electronic mail instead of or in addition to the zwrite. The following example demonstrates sending the contents of a file (/u/abc/def) into the xmail command in a ksh script to send electronic mail to userid teresia with a subject of Server Info..

Xmail command in ksh script:

cat /u/abc/def | mailx -s 'Server Info' teresia

CONCLUSION

This paper discussed many issues that are involved in creating a SAS/SHARE server, as well as considerations involved in where to locate the data and server for an application.

Creating a SAS/SHARE server is straightforward. Deciding where to put it is a bit more complicated. Many issues must be taken into consideration when going through this decision process.

Housing the data on one particular operating system does not mean the data is only available on that operating system and nowhere else. With a cross-domain
SAS/SHARE server, the data is available from other platforms as well.

Having this functionality available to you will greatly increase the possibilities for your application development. No longer are you limited to only one operating system. This allows for greater ease of data access and less worry about how to handle the future.

This has been a great asset for us and we intend to continue using it for all of our distributed applications.

The author is available for questions via email to snotia@unx.sas.com.

For further information regarding SAS/SHARE refer to the documentation mentioned in this paper.

For further information regarding remote library services refer to the Sugi Paper: The SAS System: A Complete Client/Server Solution by Cheryl A. Garner.

SAS, SAS/AF, and SAS/SHARE software are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.