Evaluating the migration of a SAS® application from a VAX® to a PC-based NT network

Alan T. Pasquino, Pfizer, Inc.
Don J. Fish, Pfizer, Inc.

Abstract: Over a period of several years, we have developed an application to handle our data entry/editing, reporting and verification as well as a variety of other data processing functions. This application was written for and ran on a Digital VAX® under the VMS® operating system. As the demands on the VAX grew, performance degraded. We therefore decided to investigate the possibility of converting to WINDOWS NT® operating system and a PC server. We also recognized other benefits including an appearance similar to other user applications; a more user friendly interface allowing use of the mouse; and the ability to share information with other PC-based applications. We wished to consider several configurations (directory locations, remote processing) to be able to determine the best system for our application and environment. This paper will discuss the configurations we decided to investigate, the characteristics we decided to measure, the method we used to evaluate each of the configurations, and the results of our tests.

Introduction: AIMS (Animal Information Management System) is our data management system written entirely in SAS. AIMS consists mainly of SAS/AF® catalogs containing SCL and PROGRAM entries. AIMS also makes use of base SAS code and FSEDIT screens. Some functions are performed by making calls to external VMS commands (i.e., creating directories, purging multiple copies of files, and setting Access Control Lists (ACL) to restrict file control of important files). AIMS is also responsible for the issuing and recording of study numbers.

Data for all studies are recorded on a group of Data Capture Forms (DCF), which are selected from a large library of forms. The user identifies the forms used by any given study, and AIMS creates a data entry menu from the list of DCFs. The user enters all data from each DCF into datasets via either PROGRAM entries or FSEDIT screens. While working on any DCF, the user has the ability to browse the data on the screen, print the data, or recall the data to make corrections. AIMS then uses PROC COMPARE to identify all data changes and writes them to an audit trail with a user/date/time stamp.

AIMS can create a report (called a datapack) containing all of the data entered for a study. The datapack is generated by running PROC PRINT or PROC REPORT for each of the DCFs used and sending the output from these prints and reports to a text file. This text file can be viewed on the screen or sent to a printer. During the data verification process, the user can generate the whole report or restrict output to a single DCF, single or group of animals, or a range of dates. This report can optionally include the statistical analysis programs written by the biometricians specifically for each study.

A data cleaning tool is also available to help check the data. This tool allows the user to write several types of computer aided checks for invalid data. The type of checks that can be performed include:

- Missing checks (identify key variables that should not be missing)
- If/then Checks (checks a certain variable based on the condition of another variables)
- Acceptable values (check a selected variable for acceptable values)
- Date checks (looks for forms on certain dates for each animal)
- Range checks or Standard Deviation (STD) checks (checks ranges of numeric values, or a 3 STD sweep).

As data are verified, AIMS can be used to lock or protect data against future editing. The user can lock an entire dataset, blocks of data or individual records. Locked data can be viewed on the screen and printed, but not changed. All records must be locked before the final report can be created.

The final report contains printouts of all data and the statisticians summary and analysis. A file is created, placed in a predetermined location and protected against deletion. Access control is then given to administrators. Also created are “Regulatory” datasets which are copies of our raw data datasets after removing any internal variables and adding several statistical variables. After the final report has been generated, no general users can access this study.

The AIMS program also includes DataBase Administrator (DBA) features to allow authorized users to unlock data (if a change needs to be made after a record has been verified) and to unlock the final report (if errors are found after the final report was created). Also included among the DBA features is the ability for programmers to write data steps and use FSVIEW screens to make corrections to existing data. These corrections are added to the audit trail so a complete history of all changes is maintained.

In addition, we have a duplicate system running at our other site in Sandwich, England. We have developed tools to install changes into a common queuing area which is copied into production nightly to ensure the two sites are running the same programs.

The initial impetus to look into other environments was the decreasing performance on the VAX as the number of users (AIMS as well as other applications) increased. We initiated several features to help improve the efficiency of the system, but as the demands on the VAX increased

Table 1. Proposed configurations

Configuration	SAS Executables	Data Directories/ Application Pgm	SAS Work Directory	Processing
1 (file server)	Server	Server	Server	Client
2 (local Work)	Server	Server	Client	Client
3 (mixed process)	Server	Server	Client/Server	Mixed
4 (terminal emulation)	Server	Server	Server	Server
5 (local SAS exe's)	Client	Server	Client	Client

and the features within AIMS increased, we were not satisfied with the performance. We had several reasons to lean towards a PC platform. All of our users already had PC's on their desk equipped with enough power for PC SAS. The increased use of PC's created a user knowledge base and familiarity with the Windows® environment. The users would be more comfortable with the keystrokes and general appearance of a Windows application. When using any terminal emulation, the keyboard of the imitated terminal has to be mapped to the PC keyboard. This leads to less intuitive keystrokes to perform certain actions. A move to a more graphical interface would allow the programmers the ability to take advantage of FRAME technology to develop screens having similar properties as other Windows applications.

Another reason promoting the PC platform, was the ability to link and share information with other PC-based applications. Many reports and forms are kept on a PC platform. With some additional coding, we could link to these at key locations. The last factor was that our final product (the final report and Regulatory datasets) eventually had to be placed on a PC platform to be electronically submitted to regulatory agencies. These reasons coupled with the strong emphasis SAS has placed on PC versions, led us to decide that the PC platform was the most logical choice.

The hardware used for the evaluation process was a COMPAQ® 5000 server with dual 200 MHz Pentium processors. The server also had 128mb of RAM and a 4 Gig mirrored disk. Four client machines were connected to the server; two with Windows 95® and 2 with Windows NT. These machines conformed to the departments' current standard desktop hardware configurations.

Configurations: We decided to test several configurations to determine the optimal setup. We settled on testing 5 configurations. The first configuration used the server as a typical file server. All SAS executables, application programs, and datasets were stored on the server. The SAS work directories were located on the server to decrease the possibility of work directories being left on client machines in the event of a system crash. The second configuration was identical to the first except the work directories were moved to the client. The third configuration kept the work files on the client and spawned the processing of the statistical analysis to the server. This configuration intended to take advantage of the power of the server, while releasing resources on the client for other applications. The fourth configuration tried to imitate the VAX setup, using a terminal emulation package. This configuration reduced the validation issues because only one machine was handling the data and doing all the processing. The final configuration was to load the SAS executables on the clients. This configuration, while increasing the installation and maintenance requirements, might increase the speed of starting AIMS and loading executables as well as reduce the traffic on the network. Table 1 summarizes the 5 configurations.

Testing Criteria: The next step was to determine the method for evaluating or comparing the different configurations. We settled on 8 characteristics that we would be interested in. After deciding on the characteristics, we defined tests or methods of measuring the characteristics that could be applied to each configuration. We decided to score each configuration on a 5-point scale (5=best), allowing that the same score could be applied to any or all of the configurations. Finally, we ranked the characteristics by importance; the ranking would determine the weight applied to the characteristic in determining a final score for the configuration. The characteristics to be measured were (ranked from most important to least important):

1. ***security*** - Each configuration would be scored on the ability to prevent unauthorized access to the data. This would include user access to the system, ability to protect data from changes, ability to lock reports (text files) from change.
2. ***validation*** - Each configuration would be scored on the ease of providing proof that the application does what it purports to do. Since the core application was the same for all configurations, the score was determined by any complexities that made validation easier or more difficult.
3. ***maintenance*** - ease of installing changes and synchronizing the application with team in Sandwich. Scoring of the configuration would include:
   - Ability to keep both locations running the same application. All changes must be installed in both the US and UK computers.
   - Ease of installing new versions of SAS or application/environment changes.
   - Steps required to recover from system/network crashes.

- Ability to add new users, new equipment or replace existing equipment (computers and printers).

4. ***speed*** - Speed would be measured in several methods. We had a pre-defined process (benchmark study) which we had used to evaluate previous changes to the system. This process included the common actions performed by the typical users, including data entry, data verification and report generation. We could easily use this process to measure the speed of the 5 configurations as well as compare back to times achieved by the VAX. To simulate the expected number of concurrent users(15-25), we decided to run the pre-defined process with 0 other users, then again with 3 and 6 other users. Using the degradation in performance, we could interpolate up to 21 users. The concurrent users were created running the report generation procedures. We chose to use the report generation for several reasons. First, the report generation is the most I/O, computer intensive process we run. Using this process for all the concurrent users, we would be overestimating the combined drain of these users. The other reason for using the report generation is that it is the most automated function we perform. There is very little human interaction, making it a consistent, repeatable action. Since we only had 4 machines connected to the server during this testing phase, we were left with 3 machines to simulate the concurrent users because one would be performing the tasks being timed. We felt that three concurrent users were not enough to try to interpolate our data to 20 users, so each machine would have to generate the equivalent of several users. We could not just run two instances of our application on a single machine and declare that to be the same as 2 users, because one client running several processes would not generate the same demand as several clients running a single process. We measured the resource utilization of a single client running a single instance of our application and compared that to the resource utilization of a single client running several instances and found that two instances were equivalent to 1.5 time the single instance. To generate the same drain as two separate machines, the client would need to run between 4 and 5 instances of our application.
   The second method of measuring speed was to run the report generation process from a true study. The benchmark study used in the time testing included a limited amount of data entry and therefore created a very small report. Reports from real-life studies are typically 100 to 1000 pages. We used data from a real study and timed the report generation. This report not only tested a more realistic report generation, but also since the report generation requires no user input, we would be testing our configurations eliminating all of the human element. We also performed this task with 0, 3 and 6 concurrent users.
5. ***resource utilization*** - The configurations were scored on several levels. Three attributes were selected and measured on both the server and client machines. We measured CPU activity(reporting the demand being placed on the CPU), network traffic (reporting amount of information be transferred between the server and the clients) and memory utilization (report the percentage of server memory being used). These measurements were made using the MONITOR application on the NT machine. These measurements were taken during the speed tests.
   Another resource test ran our application with several other applications running in the background to watch for any interactions between our application and commonly used programs. AIMS was run with WORD, EXCEL and ccMAIL running in the background.
   The final resource test ran AIMS in the background, generating a large report, and then perform standard actions in the typical user applications, such as opening, editing, printing and saving a document, receiving, opening, and sending mail.
6. ***installation*** - The configurations would be scored based on the effort required to convert the system from the current VAX application to run in the PC environment in the specified configuration. Included in this scoring would be any software/hardware requirements needed to set up the configuration.
7. ***costs*** - Each configuration would be scored on the cost of additional hardware/software required. The costs would include initial start-up costs and annual (maintenance) costs.
8. ***program modifications*** - programming changes required for AIMS to work on the new platform.

We also decided on “musts” for a configuration to be viable.

- Data Security - The configuration had to offer the same or greater level of security as the current system.
- User Verification - The configuration must be able to validate the user accessing the system
- The configuration must run all programs, all features
- The configuration must be able to maintain Groton/Sandwich Coordination
- The configuration must be able ensure Consistency of Executables
- The configuration must be able to operate on both Windows 95 and Windows NT

Conversion: We used SAS/CONNECT® to transfer our entire application and some sample data from the VAX to a PC server. This transfer was fast and easy to perform. The process to convert the code to run on the PC required approximately one programmer month. The only code that required changes was that which dealt with the

operating system. The application called external VMS commands to manipulate external files; renaming, copying, printing, and setting ACL's; this code had to be converted to an equivalent PC command where available. We also had to change all directory and filename references to reflect the PC file-naming convention. The number of changes were relatively small for the size of the application.

During the conversion, we discovered several hurdles which had to be overcome. The first problem we encountered was getting the userid; because the original application ran on a terminal based system, accessing the users identification was a simple function call. There is no similar function call in the windows environment, and no system variable in Windows 95 that contains the userid. We ended up setting our own system variable on each machine. We are currently investigating getting the user identification from the NT server.

The original application read through the log for errors. By using a batch program on the VAX to run our application, we were able to perform checks prior to and after running the SAS application. These now had to be done from within SAS.

Our reports were text files created by multiple procedures inside of a proc printto. To send this file to a printer, we executed an operating system command. We could not use a system command because it bypassed using the system drivers. We wrote a program to read the text file back in and send it to the default windows printer.

The Windows 95 clients cannot access the windows NT file ACL's to set group access to files. We therefore spawned a job to the server that ran external command to lock files.

Terminal emulation - We encountered major problems installing the terminal emulation configuration. We tried two separate packages from different vendors and were unable to get either to operate to our satisfaction. The first required a separate invocation of SAS on the server for every client running the application, which called for a tremendous amount of memory. The second package had many compatibility problems between the hardware and software. It required several third party drivers and support was not supplied by the manufacturer, but only from the vendor.

Discoveries: During the process, we also came across some unexpected benefits.

Conversion from configuration 2 to configuration 1 or configuration 3 was very minimal. In fact, configuration 2 and configuration 3 can even exist concurrently (some studies can run everything on the local machine while other studies spawn part of the processing to the server).

- While investigating methods of terminal emulation, we discovered that PC Remote was able to access on-campus machines to run AIMS from off site.
- Users have more control over the layout of their printout by allowing SAS to determine the page size from the default printer, page layout and font.
- Users control exactly which printer to go to and print to any printer that is visible to their PC.
- On the VAX, if one of the VMS commands issued during AIMS processing had an error, the batch file would not complete when the SAS program ended. With this processing moved inside SAS, this problem was removed.
- If an error was found in a final report, the report was renamed, corrected and then recreated. The new report was then compared with the old report to prove that only the desired changes were made. The programs available for the PC are more powerful than those available on the VAX.

Summary of Results: Table 2 summarizes the results of the tests.

Validation - Mixed processing (configuration 3) would require increased validation steps to prove the remote program runs properly, and that the resulting output is returned to the client properly. Using local machines for storage of the work directory (configuration 2) means the client machines need more validation. Terminal emulation (configuration 4) requires the least validation because only one machine is handling the data.

Table 2. Configuration Scores

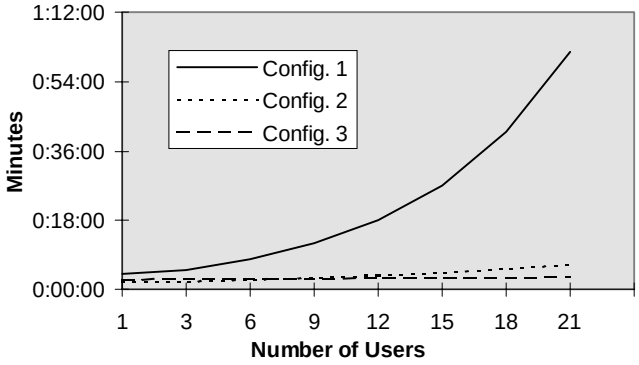
	std file server	local WORK dir	mixed processing	terminal emulation	local executables
Validation	4.5	4	3	5	3
Security	4.5	4.25	4.25	5	4.25
Maintenance	4.5	4	3.5	4	3
Speed	3.5	4.5	3.5	2 *	5
Resource Utilization	3	5	3	1 *	5
Installation	4.5	4.5	5	1 *	2
Costs	$$	$$	$$	$$	$$
Programming Modifications	4	4	4	4	4
Weighted Score	83.06	85.42	72.92	71.11	74.86

* based on experience with other applications in our computing environment.

Security - Local work directories (configurations 2,3, & 5) meant possibility of leaving copies of study data on local machines if SAS was not exited gracefully. Terminal emulation (configuration 4) would allow the highest security level (requiring userid/password at startup). Other configurations would require an open PC to be protected using some windows password system and deleting extraneous work areas on system startup.

Maintenance - Configuration 1 scored highest in the maintenance category because cleanup of the SAS work directories after a system crash is easier when located on a server rather than on individual machines. Mixed processing (configuration 3) scored lower because we must ensure the spawner is running properly. Configuration 5 was lowest due to upgrading SAS and license upgrades on individual machines

Figure 1. Speed Comparison

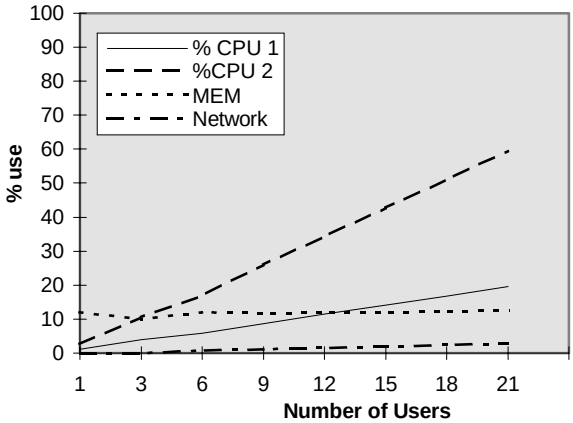


Note: 9 - 21 users projected

Speed - During the speed tests, configuration 1 showed a large degradation as users were added (see figure 1). Configurations 2 & 3 showed very little degradation. Configuration 5 was tested with 1 and 4 users and showed a 10-15% increase in speed over configuration 2. We were unable to get configuration 4 to run properly but we estimated scores based on our experiences using other applications using the terminal emulation configuration.

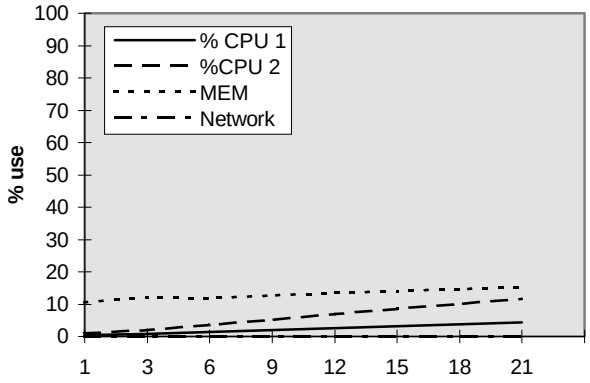
Resource Utilization *(Figures 2-4)*
The server used had dual processors and collected separate statistics on each processor. Our figures show the values for each processor (CPU 1 and CPU 2). Configuration 1 showed definite increase in CPU usage as users were added, which projected over 60% at 20 users. 60% was the threshold figure given to us by experienced network administrators as the point where performance degrades. When CPU usage rises above this value, performance is seriously affected. Configuration 3 was projected to surpass 100% at 20 users.

Figure 2. Server Utilization Configuration 1



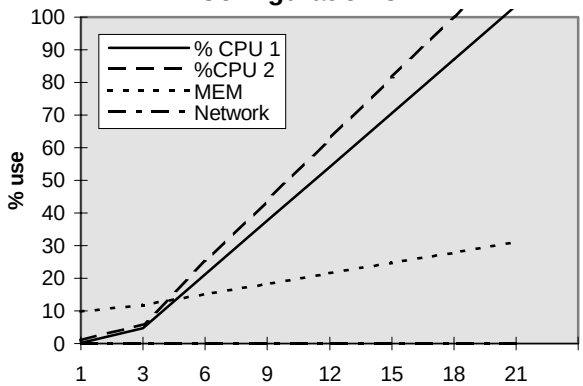
note: 9 - 21 users projected

Figure 3. Server Utilization Configuration 2



note: 9 - 21 users projected

Figure 4. Server Utilization Configuration 3



note: 9 - 21 users projected

Installation - Configuration 5 would have the most work required on every client, installing and testing SAS. Configuration 4 also would require installation of a terminal emulation software package on all clients. Configurations 1,2 and 5 might require hardware changes for any very old machines that might not be robust enough to run SAS.
Costs - All configurations had very similar server costs. Configuration 3 needs a more powerful server. Configuration 1 needs more server disk capacity. Configurations 2 & 5 means older, less powerful workstations may need to be replaced sooner. Configuration 4 requires purchase of additional software (terminal emulation package). We decided that, since the costs were going to be very similar but difficult to assess and since the cost characteristic was rated very low in our scoring system that each configuration received the same score.
Programming Modifications - The original code was converted and tested in approximately one month by a single programmer. Much of this time was spent testing the various features of our application. During the conversion from one configuration to another, we discovered very little difference between all the options. In most cases changing from one configuration to the other

could be done within a day. The only exception was configuration 4, which was going to require reformatting the server.

Conclusions: Given our application requirements, configuration 2 scored the highest. Configuration 2 had the SAS executables, application programs and data stored on the server. SAS ran on the client and created the work directories on the client. Configuration 1 (work directories on the server) was second, gaining points in validation and security, but loosing points in the performance and resource utilization. Configuration 5 (SAS executables stored on client machines) was third, having the best performance, but being more difficult to install and maintain. Configuration 3 (executing code on the client and remotely on the server) scored fourth highest. It scored low in the resource utilization. Configuration 4 (terminal emulation) scored the lowest due to the problems discussed previously.

A major advantage discovered was that conversion between configurations 2,1,3 and 5 was very simple. The difference between configuration 1 and 2 was just the location of the work directories. Configuration 2, 3 and 5 could actually coexist. If a study requires more resources than were available on a client machine, it could spawn some of the work to the server. If we decided that the increased performance outweighed the cost of installation and maintenance of configuration 5, we could simply install the executables on the client machine and change how the application is called.

As a general comparison of our benchmark times, configuration 2 showed a 60% increase in speed over our best performance on the VAX.

Acknowledgments:

Author addresses:

Alan T. Pasquino or Don J. Fish
Central Research Division - AHPD
Pfizer Inc.
Eastern Point Road
Groton, CT 06340