Paper 283-28

SAS®, Linux/UNIX and X-WINDOWS systems

Gady Kotler, Executive Information Systems Ltd., Herzlia, ISRAEL

ABSTRACT

This paper will discuss features of the native windowing system (called X-Windows) found on Linux/UNIX based systems and topics related to SAS® implementations on these systems.

As Linux/UNIX based servers and desktop systems continue to evolve, we find it important to review the different ways to implement SAS on these platforms and to compare them to other alternatives such as Microsoft based NT/2000 servers and PC's desktops or other thin clients implementations.

The paper is aimed for SAS system administrators who would like to implement an open environment within their SAS installation.

X-WINDOWS BACKGROUND

X-windows have started its way on 1984 at a project held at MIT called ATHENA. The project's aim was to create a working environment for displaying windows coming from multiple incompatible hardware and operating systems. The result was a truly open architecture that was hardware and vendor independent. The name "X" was given to this windowing system since it's the letter following "W" which was given to an earlier version.

By 1986, MIT has released for commercial use version 11 release2. Later on the development of this standard was handled over to a consortium of companies called X org. (www.x.org).

Today all UNIX vendors and Linux distributors adopted this standard for the windowing system. The current release is X11R6.6 which can be downloaded freely from the above web site.

X-WINDOWS INTERNALS

X-windows is a client-server model that separates the application logic from its GUI. This separation enables the transferring of the GUI part via a network from a host to a client.

Usually the host is a UNIX box and the client is a PC machine but not necessarily. If one uses a UNIX or Linux desktop box, then the server side and the client side are both implemented on the same box. Since X-windows is an open standard, numerous implementations exists on virtually any machine ranging from hosts like UNIX, Linux, AS/400, DEC VAX, IBM MVS/VM and clients like PC's, Macintosh, OS/2 and more.

If we implement SAS using this model as an example: the server machine is a powerful host, which is responsible for the I/O and number crunching. The client machine is responsible for displaying the SAS windows: Log, Out, Pgm, Explorer and AF windows with results coming from the server over the network.

In this context, we can use any machine with graphics capabilities residing on the network to display the application's windows as long as there is a **well defined protocol** that can handle all requests coming from the server to display and manipulate the window's contents.

X-windows is based upon such a protocol that separates the application logic running on one machine from the presentation running on another machine. These two machines are communicating between them using the X protocol.

In the X-windows world the application knows nothing about the end device that is being used to display its windows. Things like resolution, color schemes, mouse device, keyboard device etc. are left for the client machine to handle.

The client side, on the other hand, knows nothing about the application that sends him requests for displaying windows and contents inside the windows. The client side is capable to display multiple windows attached to multiple applications running on multiple servers in the network.

Special topic: Client-Server terminology

The terms "Server" and "Client" are being used in the X-windows context in a reversed manner than the way it is being presented in this paper. In X-windows terminology, the "Server" side is a piece of software running on the "small" end-user machine (The machine we usually call the client machine…).

This "X Server" package serves requests to display windows coming from "Clients" which are the applications running on the "big" machine elsewhere on the network. (This "big" machine is the machine we usually call the "Server" or "Host" machine).

Using this "Reversed" terminology, the X-windows package that is installed on a PC is called "X server" packages …

X-windows implementation is based upon several layers as follows. (Not all layers are always implemented).

1. The application.
2. The Application development standards & toolkits
3. The Xlib protocol.
4. The window manager.
5. The desktop environment.

Xlib protocol

The first layer of an X-windows implementation is a very basic protocol that is being used to transmit request to open windows and to render the application contents. This protocol has also a mechanism for sending back to the host keyboard and mouse events.

This well-defined protocol called "Xlib protocol" is the first layer of any X-windows implementation. The Xlib protocol is a truly open standard and its definition can be downloaded from X.org freely at no charge. This open source policy contributed to the popularity of the Xlib protocol and created vast implementations of devices that can handle the client side of the Xlib protocol.

Special topic: X-Terminals

Since early days of X-windows, vendors developed devices called "X-Terminals". These devices had a Monitor, Keyboard, Mouse and a very small firmware that had the X server code embedded into it and the CPU power needed to render the Xlib protocol functions. No need for an operating system, disks or other devices.

The X terminals were the first true graphical thin client machines. X terminals first appeared on the market during the late 80's (When PC's struggled to run DOS operating systems).
As the PC market evolved, and more powerful CPU's were embedded into the PC, many companies had found a way to implement the client side of the XLib protocol as an Microsoft's windows application running as a software only solution.

This trend had put an end to the traditional X terminal market (While we can still see many of them working in many sites since they were very reliable machines with no mechanical parts which practically can function for ever).

Today most of the X-windows terminals are implemented using a software solution running under MS windows (95 and above) as yet another application on our desktop. While X-windows application appear to us as a regular MS windows application, its underlying structure differs completely from any other MS windows application.

Application Development standards & toolkits

At the early days of the x protocol, no other layer was build on top of the Xlib and applications were written with direct calls to functions of the Xlib protocol. Only later, more layers were defined to simplify the development process. These additional layers include toolkits and widgets libraries, which help to standardize the look and feel of the applications.

Since Xlib protocol contains no definition of how the user's desktop or application look and feel should be, users had started to ask for a more unified desktop environment and more uniform look and feel for all thousands of X-windows applications that were written.

As X-windows became more popular and was recognized as standard de-facto of the UNIX industry, UNIX vendors began the X-windows war in order to gain control over this growing market.

Sun Microsystems was trying to establish a standard called "Open-Look" for the look and feel of applications, while other vendors fearing that Sun might "Take Over" the X-windows market with its own standards, formed a group called OSF (stands for: Open Software Foundation). The OSF group has released a competitive standard for the look

and feel of applications called the Motif Look and feel. These standards established the foundation for a common look and feel of applications with the same widget set and the same rules for designing a window, borders, menus and other objects. (Eventually, Sun has adopted Motif).

Special topic: SAS and X-windows standards

SAS uses none of the standards widget sets to implement the look and feel of SAS windows. I believe it is done in order to keep the SAS look and feel of DMS and AF windows as compatible as possible to all other SAS implementation running on other platforms.

If you'll use SAS on different UNIX/Linux platforms you'll get the same look and feel of SAS windows. The only difference you'll be able to notice is the window manager style (see below).

In SAS V6 running on Solaris there was an experimental user interface that implemented the Open Look standards but there was no continuation for this project.

Windows managers

A window manager is a piece of software that handles all events arising from displaying your application's windows along with other windows all on the same screen.

For example, when a user moves a window from one corner of the screen to another position, this is an event that is being handled by the window manager. The application knows nothing about this event. Events like moving/hiding/resizing of windows are all being handled by the window manager.

Again, thanks to the openness of X-windows, many developers around the globe have designed window managers that act and manipulate windows in many different ways. The end user can choose from many sources his preferred way of manipulating windows.

Without a window manager X-windows can be displayed on the screen but cannot be moved or resized.

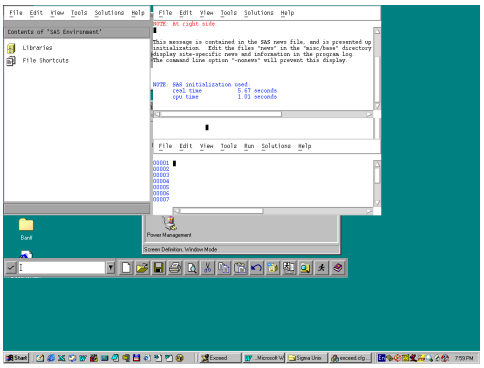

Figure 1 – SAS running under MS windows without a window manager.

As one can see from the above figure, all SAS windows appear without a border, thus cannot be removed or resized.

When working with PC's X servers, you can choose MS windows as your window manager program. This technique is mostly recommended since it saves resources on your PC and LAN and integrates your X application into one common behavior of windows.

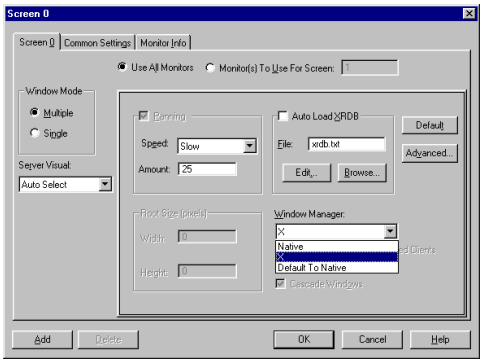

Figure 2 – Selecting native window manager (Using Hummingbird's Exceed product).

Using MS windows (Native) as a window manager means that you can use MS windows standards for resizing/moving/hiding windows the same way as you do with all other MS windows applications (e.g. MS Office applications).

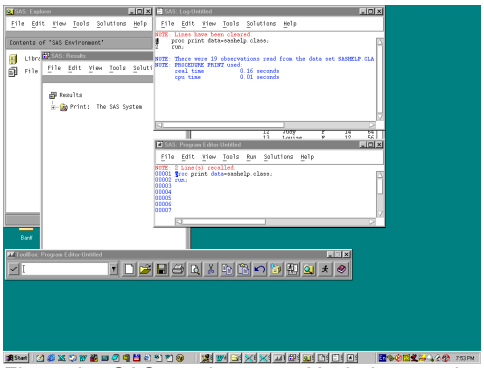

Figure 3 – SAS running as an X-windows under MS windows 98 as a window manager. (Using Hummingbird's Exceed product)

As one can see from figure 3, SAS windows appears as regular MS windows with the same borders and icons found on any MS windows application

The desktop environment

UNIX vendors and developers were also concerned about how the user's desktop should look like. Since X is an open standard, many desktops can be found on the market. Among them are the CDE (Common Desktop Environment) which is widely used on many UNIX desktop machines and KDE or GNOME, which are widely used on Linux. These desktop environments organize all your X windows in a systematic manner and let you invoke popular X applications through a series of menus.

One of the first desktop managers was Sun's "Open-Windows" which implemented the "Open-Look" look and feel:

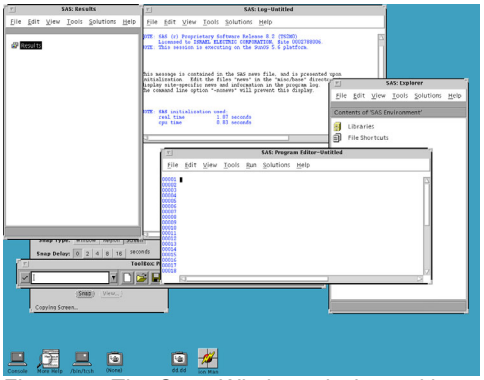

Figure 4 – The Open-Windows desktop with SAS (running under Solaris OS)

HP, DEC and IBM has promoted the CDE desktop manager which implements the MOTIF look and feel:

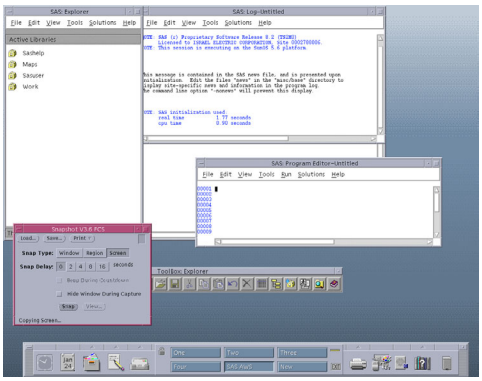

Figure 5 – The CDE desktop with SAS (running under Solaris OS)

Moving to Linux, one can see the influence of MS windows look and feel on the design of more modern desktops for Linux.

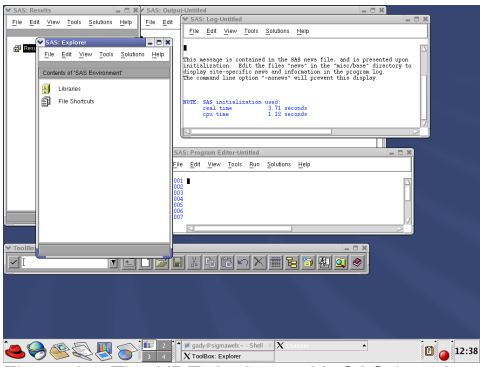

Figure 6 – The KDE desktop with SAS (running under RedHat Linux OS).

While running SAS as X-windows on an MS windows PC (As displayed in Figure 7), The desktop that is being used is the standard MS windows XP desktop. This recommended way of using your X server package help to reduce network traffic and resources used on your PC as well as unifying your working environment.

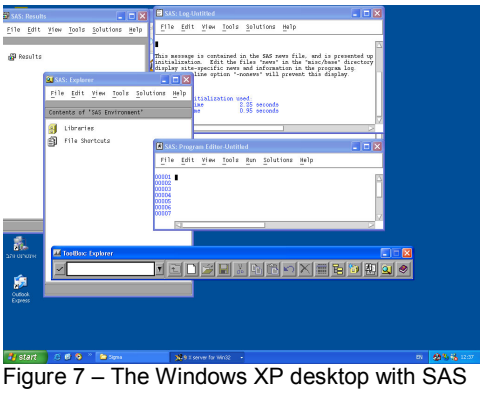

Figure 7 – The Windows XP desktop with SAS running as an X-windows application. (Using Hummingbird's Exceed product)

As one can notice from the above screen shots, the windows contents (Coming from the SAS

application) looks the same, but the windows borders look different and has different methodology for activating or resizing the window.

X-windows layers can be replaced (Except from the Xlib) to best suite the user taste and preferences.

X-windows well demonstrates two basic principals of the "Open" world of software:

1. Separate your problem into small and well-defined layers with fully published standards and API. (As opposed to bundling the OS with the GUI and keeping API's unpublished).
2. Let the users and vendors community to use imagination to implement these standards in many different ways, which enhance the user with options to choose from. (As opposed to a single one vendor solution).

To summarize the X-windows internals section lets look at the following diagram that illustrates the different layers of the X-windows system:

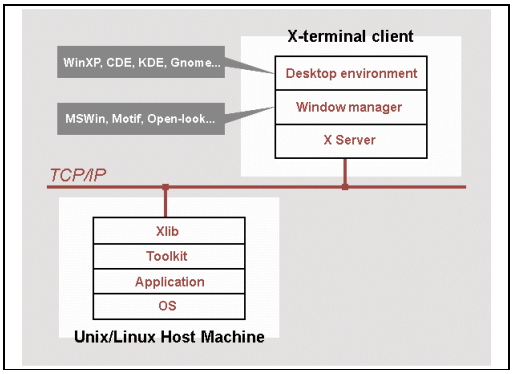

Figure 8 – X-windows layers

RUNNING SAS ON UNIX/Linux

This section will describe the numerous ways to use SAS on UNIX/Linux servers and desktops.

Thin clients implementations

By using the term "Thin client" we mean that no SAS software license is needed on the client side.

Client	Standard	Hardware/Software	Remarks/Advantages/Disadvantages
VT terminal	ASCII Terminal	An original VTxxx terminal or a terminal emulator for PC	No longer supported on SAS V8
PC	JAVA	JDMS (A SAS product with no separate license. Can be found on the SAS distribution CD)	This is the replacement for the missing VT support in V8. It is a JAVA based application running on a PC client that emulates the basic SAS DMS windows (PGM, Log , Out etc.). It has no support for SAS/Graph or AF applications.
X Terminal	X-windows	An original X terminal	A stand alone client with no integration with MS windows applications
PC	X-windows	A PC with X-server package (e.g. Exceed from Hummingbird)	A very cost-effective way of integrating the MS windows with SAS running on a server. The recommended desktop manager is WIN XP.
UNIX Workstation	X-windows	A UNIX based workstation (e.g. SUN Soalris, HP HP-UX, IBM AIX)	If your desktop is a UNIX workstation, no other software is needed since X-windows is your default windowing. Use your preferred desktop manager.
Linux Workstation	X-windows	A Linux based workstation	If your desktop is a Linux workstation, no other software is needed since X-windows is your default windowing. Use your preferred desktop manager.

SAS client-server implementations

A SAS client-server implementation is an environment where SAS modules are licensed separately both on the server side and on the client side.

Client	SAS products needed	Remarks/Advantages/Disadvantages
PC (Semi Client-Server Model)	On the Server : SAS integration On the Client : Enterprise Guide	A Visual Basic client running on your PC as an MS windows application communicating with the SAS server using a special SAS product: SAS Integration Technology. A very "Windows like" application with utilities for the novice user. On the other hand, a closed application which cannot run AF applications. Must have a license on each PC. All SAS jobs are run on the server side. The PC is only used for windowing and GUI.
PC (Full Client-Server Model)	SAS/BASE & SAS/CONNECT Both on the server side and on the client side.	A very robust environment where jobs can be run both locally on the client and on the server. Using SAS/Connect functionality, a user can split the job execution to segments running on the server for better I/O and number crunching and segments that do minor computations and graphics/reporting on the client side. Must have a special AF application to use this functionality, or be a very sophisticated user in order to take advantage of this complex environment. Requires a SAS license on every PC.
UNIX/Linux Workstation	Same as for PC client except that the user GUI is a X-windows and not a MS windows	Same as for PC client
Combined (Enhanced Client-server model)	A server-server model using "back server" and a "front server" and multiple X-windows clients attached to the "front server"	A model where functionality of splitting jobs between two servers for better I/O is used and X-windows clients are connected to the front server. This technique split overloads between many "front servers" serving many X-windows clients. Requires SAS license on each server (SAS/BASE & SAS/Connect). No need for SAS license on clients.

OTHER ALTERNATIVES

Many NT sites are using SAS PC on the desktop while storing SAS data sets on the NT server disks. This technique is wrong since the network overload while processing large data sets will be enormous. The alternatives are:

1) Use SAS client-server techniques using SAS/Connect and SAS/BASE on the server. The disadvantages of this usage are described above and are similar to the case where the server is a UNIX one.

2) Use a new concept called "Terminal Server" which is an addition to Windows 2000 servers. The terminal server uses a separation model between the presentation layer of the application and application logic while surfacing the display over the network using a propriety RDP protocol. One might think that this is similar to X-windows concept. The difference between the two systems is that X was designed originally to use a client-server model where the X server has its own part to manage windows while MS windows applications are not designed to separate these layers and the server has to do all the work.

Terminal server is a complex installation that locks you to a single vendor solution and you must have a SAS license on every PC that uses this utility. UNIX/Linux OS are designed from bottom-up to be truly multi-users OS which gives them better performance on a situation where multiple users are connected as terminals to a single server.

SAS OPTIONS FOR X-WINDOWS

This section will provide a brief discussion on how to configure SAS on an UNIX/Linux server to best use X-windows on a PC client.

Invoking SAS from your PC desktop

Every X server package provides a method to create a startup script on your PC:

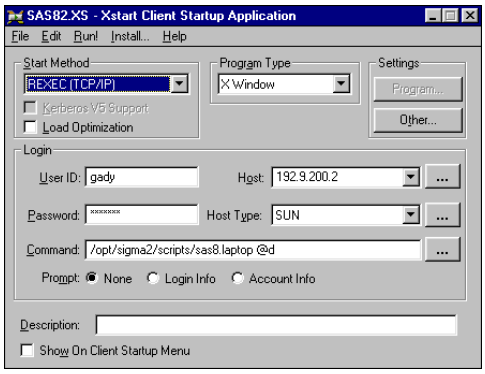

Figure 9 – creating a startup script (Using Exceed from Hummingbird).

The @d is a necessary option to any X-windows command and it is an environment variable which supplies the IP address of the PC. This value will be used as an entry for the UNIX environment variable $DISPLAY in the script file (see later) needed to run any X-windows applications.

After completing the startup script file on your PC, you can create an icon to be placed on your desktop as follows:

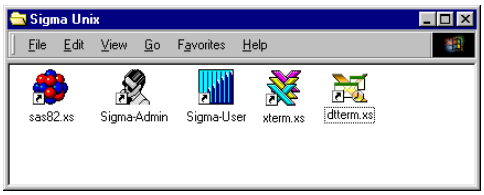

Figure 10 – Invoking SAS for UNIX with a PC icon.

Clicking on the SAS icon begins a series of actions that eventually will invoke a SAS X-windows session on your PC.

The UNIX Startup script

The UNIX startup script is a CSH UNIX file that eventually calls SAS for execution. (In the above example the UNIX startup script is */opt/sigma2/scripts/sas8.laptop*)

Its first job is to clean any previous "Zombie" processes running on behalf of this user. (A Zombie process is a process that has no terminal attached. This usually happens when a user reboots his PC without terminating SAS normally).
These Zombie processes can prevent the user from accessing his SASUSER catalogs or other data sets since they are locking them for write access.

The second task of the startup script file is to invoke SAS with the desired configuration file.

```
#!/bin/csh
#
setenv DISPLAY $1
# Killing Zombie SAS process …
set COM1=`/bin/ps -ef | grep -w $USER | grep -v
"grep" | grep "/opt/sigma2/scripts/sasv8.cfg" | awk
'{print $2}'`
if ( $#COM1 > 0 ) then
      echo "kill other sas session ..."
      kill $COM1
endif
# Invoking SAS with a configuration file …
/local/sas82/sas -config /opt/sigma2/scripts/sasv8.cfg
```

Figure 11 – SAS typical UNIX startup script

The configuration file

The configuration file controls the SAS environment. This file is being read by SAS when you point SAS exec file to it in the startup script (See above).

In UNIX you can use special configuration parameters to control X-windows behavior. These options are prefixed by *"–xrm"* (**X** **R**esource **M**anager). These *xrm* options let you control things like: Fonts, Window behavior, Desktop behavior, and many more windowing attributes. More *–xrm* options can be found in the SAS companion for UNIX documentation or in the file:

```
!SASROOT/ X11/resource_files/ Resource_Defaults
```

The configuration file points to an autoexec.sas file that contains specific SAS code to initiate your SAS environment.

Here is a sample configuration file:

```
-xrm        'SAS.startupLogo:None'
-xrm        'SAS.systemFont:    timr08'
-xrm        'SAS.DMSFont:       timr08'
-xrm        'SAS.DMSboldFont: timr08'

-sasautos ( '!SASROOT/sasautos' )
-sashelp ('!SASROOT/sascfg' '!SASROOT/sashelp' )
-autoexec /local/sas82/scripts/autoexec.sas
-maps !SASROOT/maps
-msg !SASROOT/sasmsg
-colorpgm
```

```
-sasuser     ~/sasuser.800
-work        /local/saswork

-dmsexp
-setjmp
-mvarsize 32K -msymtabmax 4M
-sortsize 48M
-memsize 100M
-maxmemquery 6M
-helploc ( '!sasuser/classdoc'
'!SASROOT/X11/native_help' )
-docloc file:/opt/sas8/install/docloc.htm
-appletloc !SASROOT/misc/applets
-news !SASROOT/misc/base/news
-path !SASROOT/sasexe
```

Figure 12 – typical configuration file

The autoexec file

This file contains SAS code that you want to be run when your session starts. It usually contains Libname definitions, Macro variables and SAS options needed.

Here is a sample SAS autoexec file:

```
/*================================*/
/* General Macro Var definitions :  */
/*================================*/

%LET DEVM    = PC2ISO                    ;
%LET KEYM    = PC2ISO                    ;
%LET ZAG     = XCOLOR                    ;
%LET GUI     = G                         ;

/*================================*/
/* PRINTING OPTIONS :               */
/*================================*/

%LET TUNXPRT = /usr/ucb/lpr -h -P        ;
%LET GUNXPRT = /usr/ucb/lpr -v -h -P     ;
%LET DEFPRT  = hp5                       ;

/*================================*/
/* SIGMA/WEB OPTIONS:               */
/*================================*/

%LET WEBIP   = 192.9.200.2               ;

/*================================*/
/* INSTALLATION DIRECTORIES:        */
/*================================*/

%let sigmadir = /opt ;   /* << === Change this !! */

libname eis          "&sigmadir./sigma2/eis_dev"    ;
libname library      "&sigmadir./sigma2/library"    ;
libname sigmall      "&sigmadir./sigma2/sigmall"    ;
libname tarall       "&sigmadir./sigma2/tarall"     ;
libname gfont0       "&sigmadir./sigma2/eis_dev"    ;
libname gdevice0     "&sigmadir./sigma2/gdevice0"   ;
libname styles       "&sigmadir./sigma2/styles"     ;

/*================================*/
/*       DO NOT EDIT BELLOW THIS LINE !!       */
/*================================*/

options nofmterr
        fmtsearch=(work sigmausr library)
        compress=yes
        ;
```

Figure 13 – typical SAS autoexec file

SAS REAL-WORLD EXAMPLES

This section describes two typical SAS installations where X-windows was selected as the preferred way of using SAS on UNIX servers.

The Israel Electric Company (IEC)

IEC has been using SAS for many years on IBM mainframes (MVS & VM). Due to heavy overload on the MVS machine and a phasing out project for the VM machines, IEC was looking for a platform substitute to its large arsenal of SAS programs, AF applications and data sets.

IEC was considering these points:

1. Need to quickly migrate SAS AF applications written for VM to the new environment without the need to rewrite them as client-server applications.
2. A client platform which support AF applications and graphics.
3. A host platform with SMP capabilities which will support hundreds of users with over 1TB of SAS datasets.
4. A cost-effective solution. Not all users are running their SAS programs concurrently, so there is no need for SAS license on every PC.

The selected solution was migrating to a UNIX server with X-windows installed on PC's. The server is a HP superdome machine with 4 CPU's, 4 GB of Ram and an HDS diskarray with disk capacity of 1 TB.

IEC has 500+ users attached to this server with an average of 25-35 users running concurrently. Plans for future improvements are to install SAS V9 with the new SPD engine in order to utilize the multiple CPU architecture more efficiently.

Gertner Institute for Epidemiology

Gertner Institute is a national research institute responsible for the health care policy in Israel. The institute employs approximately 150 staff members including researchers, statisticians and computer programmers. The institute's data base is based on SAS data sets which holds data coming from hospitals around the country and from wide range of surveys.

Currently, all users have X-windows running under MS windows 2000/XP. The server is a SUN BLADE 1000 1CPU with 40 GB of disk space and 1 GB of RAM. SAS is installed on this server with many AF/FSP data entry applications running against it.

TIPS

This section provides some useful tips that we implement in SAS UNIX projects:

1. Analyze the way users interact with SAS data sets and what procedures and application are being used in your organization. If most users do large volume data step processing and number crunching, then there is no need for a sophisticated client and X-windows is a very cost-effective solution.

2. Analyze your network performance. X-windows need a high bandwidth network. If your users are located on the WAN, then a minimum of 512 Kbps is required.

Special Topic: X-windows and Web browsers

X-windows X11R6.3 introduced a new concept for deploying X-windows application using a Web browser. This project got the name "Broadway". The X-windows applications are invoked by pointing to an HTML link and windows are displayed inside the browser as a plug-in application.

X-windows need a high bandwidth communication channel (like the 10+ Mbps LAN's). Since internet is highly dependent on low bandwidth media, The X consortium has developed the LBX protocol – **L**ow **B**andwidth **X**. This protocol uses a proxy on the server side to compress/decompress X protocol transmission.

In order to use LBX and Broadway you need a special X-windows package for your PC.

3. Use the MS Win XP desktop manager as your X-windows preferred way if you use MS windows (Or KDE/GNOME if you use a Linux box) Users will like the smooth integration of X application on their PC's.

4. Use an FTP server on your PC and write a small SAS macro to push ODS results created on the server to PC clients or directly to the company portal. In that way, users will not notice that SAS is running on a remote server, since results are displayed on their web browsers and Excel spreadsheets exactly as if they were created locally on the PC.

5. When developing AF applications for X-windows, try to write them as PROGRAM entries or FRAME entries with minimum use of images since this will overload the network.

CONCLUSION

X-windows is a stable and robust environment with benefits to SAS users. Since SAS is not highly graphical package and most of its windows display text, X-windows is a perfect companion to UNIX/Linux servers on the desktop machine.

If your installation requires hundreds of desktop machines to be connected to a SAS server, then X-windows is the perfect choice for a simple and cost-effective solution to deploying SAS on the network. Plus, you have a variety of options to suite your end users needs and taste.

REFERENCES

Many web sites have lots of X-Windows related papers. Among them I would like to note as an excellent portal for many topics discussed in my paper:

Kenton Lee Technical Window System and Motif WWW Sites:

http://www.rahul.net/kenton/xsites.html

CONTACT INFORMATION

Your comments and questions are valued. Contact the author at:

Gady Kotler
EIS Ltd.
22 Ben-Gurion St.
Herzlia, 46785 ISRAEL
Tel: 972-9-9575299
Fax: 972-9-9575305
Email: gady_k@eis.co.il
Web: www.eis.co.il

TRADE MARKS