

SAS Client-Server Technology Used for Parallel Processing of Statistical Process Control Charts

By: Randy T. Rist, Ph. D.
Target Marketing Systems
JCPenney Co., Inc.
February 28, 1997

abstract

Eight hundred (800) Statistical Process Control (SPC) X-Charts are being produced on a Compaq Proliant 586 server by a Client-Server parallel processing application written in SAS v6.12. The Compaq server has four 100 Mhz CPU's and a Compaq Raid Array.

Eight (8) clients are being executed in parallel under an NT 3.51 operating system. Clients and server communicate using named pipes. The amount of work distributed by the server to each client is determined by an exponential smoothing algorithm. The algorithm equalizes the elapse time of each client's work. Thus, starting at the same time, all clients end at the same time.

With all eight clients operating and negligible contention for resources by other applications, the throughput is 80 graphs per minute. The total number of SPC charts produced at any one time is just under 800. The total time to completion is about 12 to 13 minutes. This time measurement begins with the receipt of the data from an MVS mainframe and ends with receipt of the graphs at their storage location at a Unix web site. Total time to completion also includes about two minutes of programmed delay time.

This architecture is scalable. Although currently not necessary, provision is made for additional clients to execute in parallel on partner nodes.

Presented at the North Texas SAS Users's Group (NTSUG), SAS Institute, Dallas, Tx. on April 24th 1997

R. Rist Copy Right 1997

Introduction

Inter-nets, intra-nets, web sites, web site browsers, the Windows NT 3.51 operating system and Statistical Process Control all made their appearance at JCPenney within the same time frame. With this confluence of contemporary events, JCPenney Information Systems' management saw an opportunity to describe the time course of at first, scores and then later, hundreds of technical and managerial processes occurring in JCPenney's Information System processing centers.

A technical team was formed to integrate older MVS performance and management data bases, the older VM and Unix mail technologies, the recent Unix Web site server technology, the Windows Netscape internet browsers, and the new low cost Windows NT servers. The Statistical Analysis System (SAS) played a central role in melding these diverse technologies into a cohesive company wide management problem detection and information collection and distribution system.

After presenting a brief overview of the general system architecture that describes the role played by the SAS system, discourse focuses on how the use of SAS Client-Server technology and other key design and technical features enables the production of 800 Statistical Process Control X-Charts on a Windows NT server in 12 to 13 minutes.

General Overview of the X-Chart Delivery System

All of the data tracking the technical and management processes identified by Information System management for inclusion in the Statistical Process Control project are distributed across four central processing centers in the JCPenney network. These sites are located in Nevada, Kansas, Ohio and Texas. The data resides on MVS mainframe platforms. In these locations, some, but not all of the data are kept in SAS data sets, while other data are kept in proprietary data bases designed and/or purchased for specialized technical and/or management tasks. Data Providers, that subset of the technical team which controlled and used the data, have the responsibility of collecting, formatting and sending the data to the Texas site on a daily basis. Once there, the Data Providers, place their data in SAS data sets in a SAS daily data repository library. All of the Data Providers use SAS as part of their effort to collect, format and store data in the Texas SAS daily data Library.

At a pre-specified early morning time, a command from the Windows NT Compaq server is sent to the MVS mainframe at the Texas site to initiate a Collection, Error Rejection and Filter (CERF) application written in SAS. CERF examines the data placed in the SAS daily data Library, removes and reports on any SAS data sets and/or process data in the data sets that have not been registered with the operators or do not conform to pre-defined SAS data set and data format criteria. All remaining eligible process data series are tested for four SPC out of control conditions. The results of each process data series' SPC tests, each processes' data series, and each processes' associated web site and mail recipient addresses are transmitted to the Windows NT Compaq server. It is here that the data is processed by the SAS 4200 Graphics Engine.

The SAS 4200 Graphics Engine is a client-server parallel processing application. It is the focus of this paper. It will be examined in detail shortly. The SAS 4200 Graphics Engine server distributes the series data to between one (1) and eight (8) client directories and starts a corresponding number of clients. These clients create an X-Chart in .GIF format for each process

assigned by the graphics engine server and transmits each .GIF to the appropriate location(s) on the Unix Web Server. After completing these tasks, each client communicates with the server indicating it has completed its assignment. The server, after receiving replies from all clients, sends CERF reports, letters, mailing addresses and control information to the Unix Web Server. The graphics engine server then reports on the performance of itself and each of its clients, schedules the next initiation of the CERF job on the Texas MVS mainframe and ends execution.

Shortly after arriving at the Unix Web Server, the control information and mailing addresses are used to send reports, letters, single .GIF's and folders containing more than one .GIF to the Unix Mail Server. From there, some of the reports and letters are sent to the older VM mail system and the remainder and all of the .GIF's and folders containing .GIFS are retained and forwarded to the recipients via the newer Unix mail system.

Because the Statistical Analysis System executes on a variety of platforms, old and new, it is instrumental in integrating the old and new technologies that reside on them. In this SPC problem detection and information dissemination system, SAS is responsible for gathering, formatting, qualifying and testing data, creating graphics, and distributing the graphics and text information to both old and new mail technologies and to Web sites for access via contemporary Web browsers. SAS plays a central role in melding diverse technologies into a cohesive management problem detection, information collection and distribution system.

Adopting the Client-Server Design

The Client-Server Relationship

Client-servers are most commonly understood in the context of databases and database access. "...generally, an accepted definition describes a *client* application as the user interface to an intelligent database engine--the server." write the authors of Microsoft's *Building Successful Client/Server Applications* (Microsoft, 1997). However, commonly understood, generally accepted or not, a client-server architecture is not defined by the presence of either a database engine or a user interface. Instead, a client-server architecture is defined by two characteristics. *The first characteristic is the fact of service given and the fact of service used.* "Only applications that make use of services ... should be called *clients*. Applications ... that provide services to other applications are called *servers* " (Siyan, 1995). *The second characteristic is the presence of communication between a single server and one or more clients.* "One process [application] is defined as the *server*, while one or more other processes [applications] are defined as *clients*. In this configuration, you can have multiple clients send data to the server or the server send data to the various clients " (SAS, 1993). A design is client-server when it exhibits both of these characteristics.

Why the Client-Server Design was Adopted

The SAS 4200 Graphics Engine employs the SAS client-server technology for parallel batch construction and distribution of Statistical Process Control X-Charts. The need for client-server design is the result of performance requirements and the demand to include more and more technical and management processes under the SPC umbrella.

The SAS 4200 Graphics Engine began as cannibalized and reconstructed code from an earlier project investigating the use of Shewhart Charts for financial data. The original code was executed under HP-UX 9.01 operating system on an HP 9000/700 series work station. This code constructed .TIF formatted X-Charts and R-Charts for five financial variables for each of the 1,400 JCPenney stores. The charts were printed and bound for study. To accommodate the requirements of this new project the following changes were made:

1. R-Chart construction was eliminated.
2. The number of tests for statistical 'out of control' conditions were reduced from eight (8) to four (4).
3. A software package, Alchemey, was incorporated for transforming .TIF format graphics to .GIF format graphics.
4. A Web site directory addressing function was added.
5. A client mailing addressing function was added.

The first four changes were made very quickly. The modified graphics code was installed on a two cpu Compaq NT Server. The first 50 X-Charts were on the Web, available to the Netscape Browser and being updated on a daily basis in three weeks time. Soon the project grew to 150 charts a day, then to 635 charts and on to 800 charts. More are on the horizon.

An important requirement of the project is the delivery of either a text message or a .GIF formatted chart(s) for processes showing an 'out of control' condition to the appropriate mail recipients by 6:00 am every morning. With a demand of 50 X-Charts, processing with the reconstructed code takes about six (6) minutes. Processing takes 18 minutes with a demand of 150 X-Charts. At 800 X-Charts, processing takes one and two thirds (1.67) hours ! The 6:00 am mail requirement could not be met for most management and technical processes. To meet increasing demand and the required 6:00 am mail delivery, a new four (4) CPU COMPAQ Server with a RAID array was ordered.

Understandably, this purchase is by itself no solution. As originally designed, the old reconstructed SAS code produced all the X-Charts sequentially. Furthermore, regardless of the number of CPU's in the multi-processor, the code could only access a single CPU. Even with a RAID array, the software design limited the throughput to about 8 graphs per minute. The SAS application graphics code had to be redesigned!

A Key Re-Design Decision

The re-design began with moving the error checking and SPC 'out of control' test filter to the Texas MVS mainframe. Originally the data series error checking and out of control tests were combined in a single data step for efficiency, but this decision proved non-adaptive. The variety of data set formatting and data series errors being produced by the data provider's code and databases is well beyond what had been experienced with the earlier stores financial data. All of the newly required checks for error conditions could not be made in the single data step designed for the earlier project. The result of these changes is the multiple data step data collection, error rejection and SPC filter program, CERF, described in the prior section "General Overview of the X-Chart Delivery System."

Necessary and as important as these changes are, they still leave the major problem unsolved, "How do I use the full processing power of the four (4) CPU's and the RAID Disk Array on the COMPAQ Windows NT 3.51 server." In the JCPenney computing environment, two alternatives

presented themselves. Both involved dividing the graphics workload and processing all the workloads simultaneously. The first is sending at least four separate SAS transport Libraries and four separate requests for execution of X-Chart creation programs -- one (1) per CPU -- to the COMPAQ server. The second is sending a single SAS transport Library and a single request for execution of a server program which, in turn, initiates at least four X-Chart creation clients on the COMPAQ server.

The first alternative is rejected because it is believed that the relatively long delays, 60 to 120 seconds, possible between transmissions of the four separate SAS Libraries and requests to initiate X-Chart creation programs might not assure an even distribution of X-Chart creation programs across all four the CPU's on a busy COMPAQ server. There is a greater possibility of some of the four (4) clients running on the same CPU if the first alternative is adopted than if the second alternative is adopted. The second alternative, the client-server alternative, seemed the best alternative. It allows all of the client's to be scheduled for initiation at the same time, possibly enabling the NT operating system to distribute the clients more evenly across the CPU's. As realized, this client-server alternative is called the SAS 4200 Graphics Engine.

Technologies Supporting the Client-Server Design

Four integrated technologies enable the SAS 4200 Graphics Engine to be written as a parallel processing client-server application. These are the Server Hardware, an established JCPenney network, the Windows NT Operating System and the SAS programming system. The role of three of the contributors in enabling the graphics engine, the Compaq Server, NT operating system and SAS, is briefly presented.

COMPAQ Server Hardware

The server hardware is a COMPAQ Proliant Server with four (4) 100 Mhz Intel CPU's, 589 Megabytes of Random access memory. The controller is a Compaq Smart-2/E controller with EISA bus. It attaches to five (5) physical disks using RAID 5 logic. The total disk space is 16,375 Mega bytes. There is an Array Accelerator with cache size of 2,048 KB for read and 2,048 Kb for write. The HAL is a Compaq Proliant and SystemPro multi-processor.

Windows NT 3.51 Operating System

The Windows NT 3.51 is a multi-tasking, multi-threaded and symmetrical multi-processing operating system with integrated networking communications. These characteristics allow the NT OS to manage the resources of the computer on which it resides as well as resources across a network.

Multi tasking switches access to a CPU's resources between tasks very quickly, giving the illusion of concurrent execution. The multi threading nature of the NT operating system allows a task to actually execute two or more tasks simultaneously in different parts of the same CPU. This is concurrent execution. Symmetrical Multi Processing schedules a process or task on any one of two or more CPU's in a single computer. Therefore, two or more tasks or processes can run simultaneously on the same computer. These tasks or processes can belong to the same application.

Because Windows NT 3.51 has integrated networking communications, one Windows NT computer can also manage resources of other computers executing Windows NT. Therefore, it is possible for two or more tasks or processes to run simultaneously on more than one computer and be part of the same application.

The NT symmetrical multi-processing operating system is capable of actually executing four programs in parallel on a single computer with four CPU's. Because of its multi-tasking capabilities it is also capable of giving the illusion of executing even more. The NT networking capabilities also allow pieces of an application to run on a completely separate computer in another part of the network. As a consequence, a scalable parallel processing application can be written. All that is needed is programming software that can tap the capabilities of the NT operating system. SAS is programming software that does.

SAS Software Commands for Accessing Windows NT Capabilities

SAS software contains several commands that allow a SAS application programmer to access the integrated network communications, multi-tasking and symmetrical multi-processing capabilities of the Windows NT 3.51 operating system. These commands and the NT OS capabilities allow a SAS programmer to develop a client-server application.

Significant SAS software commands for tapping the NT capabilities and used in the design, construction and execution of the SAS 4200 Graphics Engine are:

1. **X** statement.
2. **FILENAME** statement with *device-type* set to **NAMEPIPE**.
3. **FILE** statement.
4. **INFILE** statement.

The **X** statement issues an NT command or application from within either the SAS editor or a .BAT file. In the SAS 4200 Graphics Engine, the **X** statement is used to issue the Windows NT 'AT', 'FTP' and 'COPY' commands. The 'AT' command schedules execution of .BAT files. The 'FTP' command sends files to a Unix web site. Of course, the 'COPY' copies files.

The **FILENAME**, **FILE** and **INFILE** statements can all define and/or access the Windows NT named pipe capability. A named pipe allows an application to communicate between itself and another application on the same computer or on other computers in a network.

The named pipe is a relationship between two types of pipe end points, one the *server*, the other the *client*. There is always one and only one server. There must be at least one, but there are often two or more clients. Both client and server must be defined before the named pipe relationship can be established.

Named pipes are not the only way to coordinate communication between a server and its clients, but when executing SAS client-server applications under Windows NT, it is a simple and powerful way to affect communication between a server and one or more clients.

Applying the SAS Client-Server Technology

The SAS 4200 Graphics Engine Server

The SAS 4200 Graphics Engine Server provides two services. The first is simultaneous initiation of its clients. The second is the assurance of an equitable distribution of graphics production work to each client. Here "an equitable distribution of work" means equal elapse time to completion. This assurance is provided by starting all clients simultaneously, and by distributing to each client that portion of the number of .GIFs it must complete on the basis of the exponentially weighted moving average (EWMA) rate of .GIF production. If a client typically produces more .GIFs per minute than another, it is given proportionately more .GIFs to produce. Since all clients begin execution at the same time and the amount of work they are given to do is proportionate to their rate of production, they all complete their work at (about) the same time.

In order for the SAS Graphics Engine Server to calculate the EWMA for each client, each client must communicate with the server, by giving the server its time to complete and the number of .GIFs completed. This communication is accomplished using named pipes. What follows is a block diagram outline of the Graphic Engine Server's actions showing the SAS commands executed, the Windows NT capabilities exercised and some details of the EWMA X-Chart distribution mechanism.

Server Block Design

The Graphics Engine Server design can be divided into eight (8) blocks of functions. These blocks are:

1. Set Options
2. Define Libraries and Files
3. Read and Distribute
4. Schedule Clients
5. Read Pipes
6. Distribute Instructions, Mail and Reports
7. Update EWMA
8. Schedule Next CERF
9. Print, Clear and End

Only those SAS statements that access the NT capabilities that make the Graphics Engine a client-server parallel processing application and those SAS statements that provide for equitable distribution of work among the clients have been abstracted from the Graphics Engine Server source code. The SAS statements are followed by a brief explanation of their meaning.

Set Options

OPTIONS NoXsync NoXwait;

The X statement used to access Windows NT commands, behaves differently depending on the value of related options. Two options that affect the X statement are Xsync and Xwait. Both of these options are set to 'NO'. Thus when an X statement is executed, the SAS application will not wait for the Windows NT command to be executed before continuing, but will continue with its processing. At the end of the NT command, the NT shell will not prompt or wait for the 'EXIT' command to be entered by an operator. Setting the 'X' options to *NoXsync* and *NoXwait*, allows the server and the server initiated NT commands to execute simultaneously, without operator intervention.

Define Libraries and Files

Libname

Filename

Filename <i>Tqipipe</i> NAMEDPIPE '\\rist\pipe\SPC' server retry=-1 block eofconnect byte;
--

Named pipes can be defined by a 'Filename' statement using the keyword *NAMEDPIPE*. The phrase "*\\rist\pipe\SPC*" gives the name of the physical server, indicates that this is a pipe with a file name of 'SPC'. 'Server', 'block', 'eofconnect', 'retry=' and 'byte' are keywords defining the characteristics of the named pipe.

'Server' indicates this application is the server side of the named pipe relationship. 'Retry=-1' tells the server to wait indefinitely for a pipe connection to a client. Once connected, 'block' prevents the SAS server application from processing its next procedure or data step when there are no client data available to read. 'Eofconnect' instructs the server to connect to a new client after receiving an end-of-file character from the previous client. Finally, 'byte' indicates that the data read contains no encoded record length. Thus, when the SAS Graphics Engine Server accesses the named pipe '*Tqipipe*' via a data step and INFILE statement, it will try indefinitely to connect to a client. Once connected to a client, a 'no data' condition from a connected client will not force the end of the data step. Then, when a client sends an 'end-of-file' signal, the server is instructed to connect to the next new client.

Read and Distribute

```

**A;
Data work.DistKnt(keep=ClntNmbr GifKnt);
  set syslib.SrvrHist end=mak ;          ** contains the variables Period, ClntNmbr and EWMA;
  array REWMA (64) E01-E64;
  retain NoClnt TotGif MinGif ThisPerd MaxClnt E01-E64 sumgrph 0 ;
**B;
  If _N_ = 1 then do;
    **Read Macro Variable Section;
    TotGif=symget('TotGif');
    MinGif=symget('MinGif');
    ThisPerd=symget('ThisPerd');
    MaxClnt=symget('MaxClnt');

    **Decision Rule;
    NoClnt=floor(TotGif/MinGif);
    If NoClnt = 0 then NoClnt = 1;
    else if NoClnt > MaxClnt then NoClnt=MaxClnt;

    **Create Macro Variable Section;
    call symput('NoClnt',NoClnt);
    end;

    **Total # X-Charts;
    **Min # Xchats / Client;
    **Current Processing Period;
    **Maximum # Clients allowed;

    **# Clients Activated;

**C;
  If Period = ThisPerd then do;
    N+1;
    if N <= NoClnt then do;
      REWMA(N)=EWMA ;
      sumgrph=sumgrph+EWMA;
      delete;
    end;
    else delete;
  end;

  else delete;

**D;
  If mak = 1 then do;
    do i = 1 to NoClnt;
      GifKnt=floor((REWMA(_i_)/sumgrph)*TotGif);
      SgifKnt=SgifKnt+GifKnt;
      if i =NoClnt then do;
        GifKnt=GifKnt + (TotGif-SgifKnt);
      end;
      output work.DistKnt;
    end;

    **Sum of .GIFS Distributed;
    **on last;
    ** add underage to last;

  end;
end;

```

X 'FTP -v -d -n -i -s:Drv:\trnsport.dat Ip_Address ';

The data step illustrates how the EWMA is used in conjunction with other necessary information to determine the number of clients that are needed to handle the workload, and how many .Gifs --

X-Charts -- are to be created by each client.

In section '**B;' the code calculates the number of clients the server will initiate to handle the queued workload. The decision rule includes constraints. As the Graphics Engine is configured today, no less than one (1) client and no more than the preset maximum of eight (8) clients can be initiated.

Historical data for each of the maximum number of eight clients are stored in 'syslib.SrvrHist', section '**A;'. For each of six (6) daily time periods, each client's rate of .GIF production -- EWMA -- is stored. Code in section '**C;' selects this information for the period in which the server is currently operating. The number of .GIF's produced per unit time --EWMA-- is placed in an array (REWMA(*i*)). The total number of .GIF's produced per unit time by all clients about to be activated is calculated (sumgrph).

The array elements and total are used in section '**D;' to calculate the number of .GIF's each client is to process. Because the *floor()* function is used to round down to the nearest integer, the sum of the total .GIF's distributed will almost always be less than the total number of .GIF's queued. This underage is added to the last client's workload.

There are obvious opportunities to improve the distribution of the workload to the clients. One is to collect EWMA data for daily periods and days of the week for each client, not just daily periods. Another is to redistribute the underage to all of the initiated clients, not just the last.

After the distribution calculations are completed, the server distributes the raw data to SAS libraries in client directories on the Compaq server. That data that is destined for client directories on partner nodes is stored locally in transport format and then transmitted using the X statement to initiate the NT 'FTP' command shown above. The details of this command will be examined shortly.

Schedule Clients

X 'AT \rrist 5:30 "Drv:\client01.bat" ;

After distributing the data, one NT 'AT' command is issued by one SAS X statement for each client required by the work load. The NT 'AT' command places a request to execute an NT command in the NT Scheduler que. The time of execution is assigned using military time (24 hour clock). The command above requests execution of a .bat file at 5:30 am. A client may be executed on the resident computer 'rrist' or executed on a partner node by changing the computer name after the double back slash (e.g. '\rneil').

Read Pipe

```

Data work.clients(keep=NoClnt ClntNmbr StrtTme EndTme GifKnt Period);
INFILE Tqipipe length=len missover,
input @;
retain NoClnt 0;
if _N_=1 then NoClnt=symget('NoClnt');
if len ne 0 then do;
input @1 ClntNmbr $ 8. @10 StrtTme 12.0 @24 EndTme 12.0 @ 38 GifKnt 8.0
      @46 Period 2.0;
N+1;
if N = NoClnt then do;
output work.clients;
stop;
end;
else do;
output work.clients;
end;
end;
else do;
end;

```

The data step shown above is used to listen for client's messages soon after the clients are queued in the NT scheduler for initiation. Each client sends a single message to the server upon completion. This contains its client number, its start and end times, the number of X-Charts it created, and the period in which it began executing. Once the server data step reading the pipe receives one message from each client, the data step stops. No more client data is stored in 'work.clients' and the next server activity begins.

Distribute Instructions, Mail and Reports

```

X 'COPY Drv:\TempDir\mailrpt.txt Drv:\OcDir\mailrpt.txt';
X 'FTP -v -d -n -i -s:Drv:\mailall.dat Ip_Address';

```

After receiving the indication from all initiated clients that they have constructed and transmitted their X-Charts to their assigned locations, the server distributes mail and reports to a mailbox on the Unix Web site along with instructions used by the .cgi's for displaying the 'out of control' X-Charts on the Web. The server is written so that if the mailbox directory is located on the Compaq, X statements issue 'copy' commands to move the text format mail, reports and instructions from a temporary server directory to a Web out-of-control directory. If the mailbox directory is located on another machine--a Unix box--as it is at JCPenney, then the X statement issues an 'FTP' command which transports the text information.

The 'FTP' command has the following toggles set:

-v	verbose	-- full comments are displayed
-d	debug	-- debugging comments will be displayed.
-n	no auto-logon	-- initial connect logon is disabled.
-i	interactive prompting	-- interactive prompting during multi-file transfer disabled.
-s:	command file follows	-- execute FTP commands contained in the file referenced.

Thus the command, 'FTP -v -d -n -i -s:Drv:\mailall.dat Ip_Address ', says:

1. Inform me,
2. Don't prompt for a logon at initial connect.
3. Don't prompt during multiple file transfers.
4. All the information you (Ip_Address) need for logon etc. is contained in the referenced file.

Update EWMA

```
Data syslib.SrvrHist(keep=Period ClntNmbr EWMA);
merge syslib.SrvrHist(in=H) work.clients(in=C); by Period ClntNmbr;
if H=1 and C = 1 then do;
  EWMA= (.8*EWMA)+(.2* (GifKnt / (EndTme-StrTme) );
end;
```

The syslib.SrvrHist data set is then updated with the newly calculated exponentially weighted moving average. The EWMA is calculated for each client. Only the appropriate period is updated. The two term update expression contains two weights, '.8' and '.2'. These two weights must sum to unity (1.0). Beyond this constraint they may vary, depending upon the wishes of the server operators. As chosen in the SAS 4200 Graphics Engine, the constants say that the historical performance of the client is four (4) times more important than the client's most recent performance in estimating its performance next time.

Schedule Next CERF

```
X 'AT \rrist 13:30 "Drv:\CERFSchd.BAT" ';
```

The X statement is used to issue an NT 'AT' command to schedule the next initiation of the CERF job on the Texas MVS mainframe.

Print, Clean and End

```
proc print data=work.perform label split='_';  
Title 'Server Performance Data';  
proc delete data=work._all_;  
proc delete data=templibs._all_;  
endsas;
```

The clean and end section deletes all SAS data sets from the server's work and temporary SAS server libraries. The 'endsas;' command then terminates the server job.

The SAS 4200 Graphics Engine Client

The SAS 4200 Graphics Engine Client's work is the creation and delivery of X-Charts to their home and out-of-control libraries on a Web Unix server and reporting its performance and successful completion to the Graphics Engine Server.

Client Block Design

The Graphics Engine Client design can be divided into six (6) blocks of functions . These blocks are:

1. Set Options
2. Define Libraries and Files
3. Read and Plot X-Chart Data
4. Distribute X-Charts
5. Write to Pipe
6. Print, Clear and End

As with the Server Block Design, only those SAS statements that access the NT capabilities that make the Graphics Engine a client-server parallel processing application and those SAS statements that provide for equitable distribution of work among the clients have been abstracted from the Graphics Engine Server source code. The SAS statements are followed by a brief explanation of their meaning.

Set Options

```
OPTIONS NoXwait NoXsync;  
GOPTIONS DISPLAY GACCESS=GSASFILE GSFMODE=REPLACE ;
```

As indicated in the Server Block Design, setting the 'X' options to *NoXsync* and *NoXwait*, allows the server and the server initiated NT commands to execute simultaneously, without operator intervention.

The default graphics options set the environment for placing output from SAS graphics procedures to be placed in a SAS graphics catalog. However, when creating graphics for storage and use by systems other than the SAS system, the Graphics Stream File (GSF) is the method of choice. The **GOPTIONS** statement above sets the graphics options to produce a GSF in an external file that must have the file reference name of **GSASFILE**. The **GSFMODE** option set to **REPLACE** indicates that each time a graphics procedure writes to the external file referenced by the name 'GSASFILE', the previously produced graph will be over-written.

Define Libraries and Files

```
Libname .....  
Filename .....  
Filename GSASFILE 'Drv:\clntpath\temp01.gif';  
Filename Tqipipe NAMEDPIPE '\\rrist\pipe\SPC' client retry=-1;
```

Two files of note in the Define Libraries and Files section are referenced by the file references *GSASFILE* and the *Tqipipe*. The file reference, *GSASFILE*, is a name reserved for use by SAS graphics procedures as a reference to an external file that is a repository for Graphics Stream File format graphics.

The file reference, *Tqipipe*, on the last Filename statement above, references the same named pipe as the server Define Libraries and Files section references, '\\rrist\pipe\SPC'. It is this common named pipe reference that makes the server and the client(s) part of the same, larger application. The keywords for *Tqipipe* are limited to 'client' and 'retry='. Unlike the server named pipe, the keywords block, byte and eofconnect are not present. 'Block' and 'byte' are defaults. 'Eofconnect' applies only to the server.

The client will try to connect to the named pipe, indefinitely.

Read and Plot X-Chart Data
<pre> Data work.RawPart; fp = input(left(symget('FPOINT')),8.0); mxsrln = input(left(symget('Mxsrlen')),8.0); do fpoint = fp to (fp+mxsrln); set work.RawAll point=fpoint; end; </pre>
<pre> Proc Gplot data=work.RawPart gout=GSASFILE; </pre>
<pre> X 'COPY Drv:\clntpath\temp01.gif Drv:\clntpath\GifName.gif' ; </pre>

The steps in the Read and Plot X-Chart Data block are executed inside a macro loop. The loop is iterated until the last set of process data is transformed into an X-Chart and copied from the temporary file referenced by the GSASFILE keyword, to a uniquely named .GIF. The SAS X statement accesses the NT 'COPY' command at the bottom of the loop.

An important contributor to the speed at which the clients produce graphics charts is the use of the **point=** option on the **set** statement. This provides a means of directly addressing the location of the first observation in each process's series without reading all prior or all subsequent observations in the '*work.RawAll*' data set.

Distribute X-Chart
<pre> X 'COPY Drv:\clntpath\GifName.gif Drv:\HomeDir\GifName.gif' ; X 'COPY Drv:\clntpath\GifName.gif Drv:\OcDir\GifName.gif' ; X 'FTP -v -d -n -i -s:Drv:\SendGif.dat Ip_Address' ; </pre>

The Distribute X-Chart block uses the X statement to access NT 'COPY' commands to move the X-Charts from the client path to their home and/or out-of-control directories on the COMPAQ server, or to access the NT 'FTP' command to move the X-Charts from the client path to their home and/or out-of-control directories on the Unix Web site.

Write to Pipe

```
Data work._null_;  
  file Tqipipe;  
  set work.clientme;          **contains ClntNmbr, StrtTme, GifKnt and Period;  
  retain endTme 0;  
  endtime=datetime();  
  put @1 ClntNmbr $ 8. @10 StrtTme 12.0 @24 EndTme 12.0 @ 38 GifKnt 8.0  
    @46 Period 2.0;  
;
```

The client number, start time, end time, number of X-Charts created and the time period in which the client creates the .GIF's is sent as a single observation over the named pipe referenced by the file reference 'Tqipipe.' To get to the server that initiated the client, 'Tqipipe' must reference the same physical reference (i.e. \\rist\pipe\SPC) as the server file reference, 'Tqipipe', references.

Print, Clean and End

```
proc print data=work.clntinfo label split='_';  
  Title 'Client Performance Data';  
  X 'DELETE Drv:\clntpath\GifName.gif ' ;  
proc delete data=work._all_;  
proc delete data=templibs._all_;  
endsas;
```

The print, clean and end section prints a client performance message and deletes all X-Charts from the clients directory via the SAS X statements invocation of the NT 'DELETE' command. This block also deletes all SAS data sets from the server's work and temporary SAS Libraries. The 'endsas;' command then terminates the client job.

Summarizing this Application of the SAS Client-Server Technology

The SAS system provides programming statements that access NT operating system capabilities. The programming statements and the operating system capabilities they access make writing a linearly scalable client-server parallel processing application tractable. The FILENAME, INFILE and FILE statements allow the SAS application programmer to define and use named pipes for communication between a server and its clients. The X statement is used to access the NT 'AT' command that can schedule and initiate processes on a server computer node and on any number of partner nodes. The X statement is also used to access the NT 'FTP' command that transports data from the server node to various other partner nodes in the network. Thus a client-server application can be linearly scalable because the number of clients that may be initiated simultaneously can be linearly increased with the addition of nodes.

The value of parallel processing is completing a task of a given size in a shorter amount of time than would be possible with serial processing. In a situation in which a deadline or window can not be met, breaking an application into pieces and running the pieces simultaneously is a natural solution.

The NT 'AT' command allows all clients of a parallel processing client-server application to begin simultaneously. This solves only half the problem. In order to receive the greatest reduction in elapse time from a parallel processing application, all cohort clients must *begin and end* at the same time. However, there is no NT capability that can assure that the work among the clients is 'equal' or that all clients will end at the same time. This capability must be programmed within the application by the SAS programmer.

Distributing the Work, A Programmer's Choice

There are many ways of looking at the concept of work on a computer. Amount and kind of resources used is a fairly common way to construe the meaning of 'work.' In this case the number of GIFs constructed could be useful. What ever is used to measure and distribute the work among clients, it is the application programmer's choice. It should, however, reflect the goals of the application and the conditions of the application's environment.

The goal of the SAS 4200 Graphics Engine is clear, reduce the elapse time required to produce an X-Chart. Examination of the potential sources of elapse time elongation in a scalable parallel processing application resulted in a quick list. It was clear that many of these sources of elongation would have an approximately equal affect on all clients executing in parallel. It was also clear that there were some sources that had the potential for creating systematic performance differences among the simultaneously executing clients. Immediately evident is choosing a partner node on which to execute additional clients that has a computing capacity which is more or less than that of the server node. When compared with server node clients, differences in a partner node's CPU speed, memory size, paging, bus speed, disk configuration and utilization can have a profound systematic affect on the performance of those clients that execute there. Transmission of the X-Chart output across different paths in a network can have a similar differential effect. The performance of the client will vary systematically with the time of day. And not so obvious at first, are daily differences in the additional incidental amount of processing done by each client. (This incidental processing is a function of the number of X-Charts that show 'out-of-control' conditions. After operating the application, it came to be known that the proportion varies systematically by day of the week.)

The method of distributing work among clients must adjust for the affects of systematic differences when they exist and ignore them when they don't. It was decided that each client should record its own performance. Thus, the effect of any event that systematically alters the performance of one or more clients will be recorded and correctly associated with the appropriate client. Because time of day is known to affect performance, client performance is associated with the client and also the time of execution. Thus in the case in which there are clients operating on two nodes of equal capacity, the first node with a light load all day and the second node with a light to heavy load with the heavy load occurring at the same peak processing time each day, the clients on both nodes record equally strong performance during the light periods, but the clients on the second node record degraded performance during the peak processing time.

Clients record performance as a rate of X-Chart production - .GIFs/min. The lower the performance the lower the rate, the higher the performance the higher the rate. By distributing work to clients such that the clients with lower rates of production receive proportionately fewer graphs to produce and the clients with higher rates of production receive proportionately more graphs to produce, each client can be made to complete its work in the same amount of time. For a given number of clients operating on different capacity computers or computers with similar capacity and varying workloads, this method of work distribution enables the application elapse

time to be kept at a minimum.

A Numerical Example Distribution Proportional to Rate Formula

Let's illustrate. Two clients, one on the first node and one on the second node, both execute during a light processing period. They each record a performance of five (5) X-Charts completed every minute. Tomorrow during the same period they must collectively produce 12 X-Charts. Tomorrow each client will be given an equal number to produce because yesterday's performance data indicates equal performance. Numerically :

No Charts for Nth client = ((Nth client's rate)/(client1's rate + client2's rate))*Total No X-Charts

$$\text{No X-Charts for client1} = ((5/(5+5))*12)=6$$

$$\text{No X-Charts for client2} = ((5/(5+5))*12)=6$$

Let's assume the same two clients, client1 on the lightly loaded node and client 2 on the heavily loaded node. Yesterday's processing performances being five (5) X-Charts per minute for client1 and three (3) X-Charts per minute for client 2. Proportionate distribution of one hundred sixty (160) X-Charts to each client on the basis of prior performance results in one hundred (100) X-Charts being distributed to client1 and sixty (60) X-Charts being distributed to client2.

$$\text{client1} = ((5/(5+3))*160)=100$$

$$\text{client2} = ((3/(5+3))*160)= 60$$

With stable systematic performance differences between the clients and the same rate for each today as yesterday, both clients will complete their task in twenty minutes ($100/5 = 60/3 = 20$). With stable systematic performance, this simple formula works well. However, strong non-uniform and unsystematic transient events limit the value of this simple algorithm.

Let's say that a new application is installed on the same node as client1. It has been improperly tested and is causing a system wide performance problem. Yesterday client1's performance was five(5) X-Charts per minute, and today, as indicated above, it is assigned one hundred (100) of one hundred sixty (160) charts to produce. Yet, today client1 is slowed to a rate of three (3) X-Charts per minute. While client2 completes its sixty (60) X-Charts in twenty minutes, client1 completes its one hundred (100) X-Charts at its new rate in thirty three and one third (33.3+) minutes. This elapse time is a full 66% longer than that of client2 and a full 66% longer than the best possible elapse time. Worse yet, the problem is now carried into the future. Here is how.

Client1's most recent performance rate is now equal to client2's, 3 X-Charts per minute. Tomorrow 160 X-Charts are again distributed between the clients, but now each client receives an equal number of X-Charts to complete, eighty (80). Yet, because the badly written application no longer executes on the node, client1 will process the X-Charts, not at yesterday's three (3), but at its usual five (5) per minute. Client1 will complete its eighty (80) graphs in 16 minutes, while client2 will complete its eighty (80) graphs in 26.6 minutes. This is a 10.6 minute difference and it is a full 33% longer elapse time than the best possible elapse time of twenty (20) minutes. Client1's previous extra long elapse time (13.3 minutes) has been translated by the simple distribution formula into today's extra long elapse time (6.6 minutes) for client2!

Using the simple distribution formula above, unsystematic transient influences effecting some but not all clients can greatly reduce a parallel processing application's effectiveness in shortening elapse time. But there are fixes.

Exponentially Weighted Moving Average

There is no way to diminish the effect of an extreme transient event on a client when it is occurring. However, there are ways to diminish today's extreme affect on tomorrow's work distribution. One of them is by substituting a weighted average of the client's historical and current performance for the daily arithmetic average in the distribute in proportion to rate formula. One approach is called the Exponentially Weighted Moving Average (EWMA).

In our previous example, client1, on a lightly loaded node, typically produced five (5) graphs per minute, while during the same period, client2, on a highly loaded node, typically produced three (3) graphs per minute. These typical rates are now our historical rates. The historical rate is identified by the variable **HISTRATE**. The performance rate calculated by the clients each time they execute is now our daily rate. It is identified by the variable **DAYRATE**.

To diminish the effect of today's transient influence on tomorrow's distribution of work to the clients, the historical rate and current rate are weighted and averaged to produce the value of **EWMA**. The weights sum to 1.0. They are chosen to emphasize the relative influence of historical and daily performance.

where:

$$0 \leq w_2 \leq 1$$

$$w_1 = (1 - w_2)$$

$$\text{EWMA} = (w_1 * \text{HISTRATE} + w_2 * \text{DAYRATE});$$

If w_2 is set equal to 1 then w_1 is set to 0 and the historical or typical rate has no influence on our **EWMA**. This is equivalent to the arithmetic average used earlier. Consequently, today's rate will influence tomorrow's distribution in precisely the same way. The full effect of a strong transient today will be passed on to tomorrow's distribution of work. To avoid this and dampen the future effect of a strong transient, w_2 is reduced in magnitude. The historical rate begins to exercise more influence, the daily rate less. Acting as a mathematical fly wheel, the historical rate stabilizes the daily distribution of work to the client. Here is how.

In the example of the transient diminishing of client1's performance above, client1's rate of graph reduction dropped from five (5) to three (3) graphs per minute one day, and the next day, after the node had return to normal processing, the client1 was distributed fewer graphs than usual, eighty instead of one hundred. Client2 was distributed more than usual, eighty instead of sixty. The result is that the clients finish their work 10.6 minutes apart. The application elapse time is 26.6 minutes and 33% greater than the best possible elapse time of 20 minutes. These results are the same as would be obtained if the **EWMA** was calculated using $w_1=0$ and $w_2=1.0$. Let's emphasize history in this new example by setting $w_1=.8$ and $w_2=.2$.

$$\text{EWMA client1} = [(.8 * 5) + (.2 * 3)] = 4.6;$$

$$\text{EWMA client2} = [(.8 * 3) + (.2 * 3)] = 3.0;$$

EWMA is then substituted for the rate values in the distribution formula. As a result, each client is now distributed the following number of X-Charts to produce.

$$\text{client1} = ((4.6 / (4.6 + 3.0)) * 160) = 97$$

$$\text{client2} = ((3.0 / (4.6 + 3.0)) * 160) = 63$$

With the true production rates remaining at 5 and 3 for clients one and two respectively, the

actual elapse times are 19.4 minutes and 21.0 minutes. The difference in elapse time is reduced to 1.6 minutes (76% reduction). The application elapse time is 21.0 minutes and only 5% greater than the best possible elapse time of twenty (20) minutes. The fly wheel effect of the EWMA prevents the distribution by proportion to rate formula from over compensating for extreme transient changes in client performance. Depending on the weights, EWMA provides a stable rate.

From this example it might be argued that the stability of the formula provides too stable a rate. What happens if the so called 'extreme transient' is the first in a series of low performances that are here to stay? Perhaps instead of a transient program error, the cause of client1's decrease in performance is the introduction of a new workload that represents a constant competitor for resources. Then the new typical performance rate for client1 is going to be three (3) graphs per minute not the former five (5). Unless the HISTRATE value is updated to reflect the new typical rate, client1 will continuously receive 97/160 X-Charts. It will continuously take $97/3 = 32.33$ minutes to complete. And the application elapse time will always remain 12.33 minutes longer than its best possible elapse time of twenty (20) minutes.

Fortunately, the formula for the EWMA takes this into account. Once calculated the EWMA is now the updated historical rate because it is the average of the past and the current. Tomorrow it is history. So the formula is coded as:

$$\text{EWMA} = (w1 * \text{EWMA} + w2 * \text{DAYRATE});$$

If the typical rate changes from five (5) to three (3), then the recorded DAYRATE will, time after time, be three (3). This adjusts the EWMA downward a step at a time until it equals the DAYRATE. A new equilibrium is established. The elapse time of the total application is at its minimum.

Summary of Key Design and Technical Features

An X-Chart graphics production system originally produced 8 graphs per minute or 480 X-Charts in .GIF format per hour. At this rate the full complement of eight hundred (800) graphs takes 100 minutes to complete and mailing deadline could not be met. The application is re-designed and is ten (10) times faster, completing the eight hundred (800) graphs in 10 to 12 minutes. The key design features of the new code is the client-server concept, defining 'equal distribution of work' as 'equal time to completion' and scalability. A single server distributes work to its clients. The clients start at the same time, work in parallel and then end at the same time. The throughput of the client-server application can be scaled upwards.

Key technical features supporting this design are:

1. A communications network connecting computer nodes, the JCPenney network.
2. Computers with symmetrical multi-processing and multi-tasking capabilities, the COMPAQ Server with RAID array.
3. A networking operating system with the capability to schedule and equitably distribute processes over multiple CPU's and nodes, the capability to allow one process to communicate with other processes, and the capability to transfer data within and between nodes, Windows NT 3.51.

4. The software able to access the operating system's capabilities in a simple and effective way, SAS.
5. A method of distributing work to equalize elapse time among clients, EWMA and Distribution Proportional to Rate formulae.

All of these features are integrated in a working client-server parallel processing application that is ten times faster than its predecessor. The application's throughput is capable of being scaled by adding additional computer nodes to handle client processes. The name of the application is the SAS 4200 Graphics Engine. It was designed and built at JCPenney.

REFERENCES

- Box and Jenkins Time Series Analysis: Forecasting and Control,
Holden Day, San Francisco, Ca. 1970.
- Chatefield, C. The Analysis of Time Series: Theory and Practice,
John Wiley & Son, N.Y., 1976.
- Microsoft "Building Successful Client/Server Applications", 1997
<http://www.microsoft.com/visualtools/scenarios/clientserver.htm>
- SAS SAS Companion for the Microsoft Windows NT Environment,
SAS Institute; Cary, NC, 1993.
- Siyan, Karanjit Windows NT Server, New Riders Publishing, Indianapolis, In., 1995.

CONTACTS

Contacts at: JCPenney
Dallas, Tx.

Lee Freeman - Manager AITTS/Planning and Support Group
Brenda Greathouse - HTML and daily operation.
Ron Kveton - Statistical Process Control Charts