SAS AS A SYSTEM PROGRAMMER'S TOOL

James D. Oleksiw
Northeast Utilities Service Company

ABSTRACT

The SAS language is very rich in its capabilities to read and analyze data. These features of the language make it a very desirable tool for the systems programmer. Most of the uses of SAS in systems programming have been in the area of SMF/RMF data reduction. There are many other applications for SAS. This paper will present a few of these applications.

PROBLEM: SMP REPORTS

IBM's Systems Maintenance Program, better known as SMP, provides an excellent application for SAS. SMP reports are long and it is difficult to obtain a concise report organized in a logical manner. Northeast Utilities Service Company (NUSCO) desired a brief summary of PTF's on its system, ordered by PTF number and also by date. The SAS data handling capabilities also allow listings of all SYSMODS applied and not accepted; SYSMODS received and not applied; PTF's that have been superceded; and PTF's applied for each FMID.

The method of generating this report is quite simple. First, an SMP run executes a LIST CDS. The output of the SMP is saved on a temporary disk dataset. SAS then reads the temporary dataset and inputs the pertinent information. The procedure to run this job and a sample output are contained in Figures 1 and 2.

PROBLEM: ACF/VTAM NETWORK DESCRIPTION

Management of large communications networks has become more difficult with their increasing size. NUSCO experienced problems in identifying terminals with their locations; serial numbers, owners, cables and controllers. For example, several devices in the same room might be interfaced to different communications controllers, lines, modems, and 3705 controllers. The cables to the terminals might run between floors and be difficult to trace. No inventory report was available to indicate makes and models of terminals. With this in mind, a SAS application was developed to maintain a database of terminal information. This database contains terminal serial number, access method (VTAM) id, building, room, operator, manufacturer and model, channel address, and cable number. SAS easily manipulates the information to provide reports by serial number, VTAM id, counts of terminals in each building and room, and terminals on each 3705. A list of spare devices is also maintained in this database.

This SAS database is valuable in daily operations, as it provides computer operators with a quick cross reference of access method names with physical devices. This reference also helps in checking the accuracy of the list, as terminals have a habit of changing cables, walking across rooms, and remanufacturing themselves.

Validation of monthly bills is aided by checking billing by serial number against our database records. CICS terminal names will be added as CICS comes online at NUSCO. As with any collection of data, it is imperative that any changes in the physical world be reflected in the logical world. With this in mind, the operations personnel responsible for hardware maintenance are also responsible for database maintenance.

A sample report from this SAS Terminal Database is shown in Figure 3.

PROBLEM: VTAM/TSO BUFFERSIZE

TSO under VTAM performs best when the size of the TPUT/TGET buffers is known. There is no direct way to measure and analyze these buffers. A "good" length for the VTAM TSO (TCAS) VTIOC buffersize specified in SYST.PARMLIB, TSOKEY90, will yield better TSO performance. NUSCO could not perform an accurate estimate of the BUFRSIZE parameter since there is such a wide variation in applications and terminals under TSO. The network consists of a mix of 3277-11's, 3278-2, 3278-4, 3278-5, and teletype devices.

NUSCO TSO users make use of the IBM product, TSO Session Manager; there is also a variety of other full screen applications; the rest of the network uses "normal" TSO commands. The major performance problem is to pick the size of a buffer that would contain roughly ninety percent of all TPUT/TGET requests.

SAS coupled with GTF solves this problem. GTF records the results of an SVC 93 (TGET/TPUT). This trace produces thousands of records an hour. The only meaningful way to wade through the vast quantity of data is to have SAS read in the output of AMDPRDMP and then analyze the results. With this technique NUSCO learned that its average TPUT/TGET buffer size is over 200 bytes. PROC FREQ showed, however, that there was a tendency to do mostly very short terminal I/O; ninety percent was less than 132 bytes, with the remaining ten percent very large buffer requests, 4K or more bytes.

This method used AMDPRDMP to initially read the GTF dataset. SAS could undoubtedly read this directly and more efficiently; however, the author had not yet found a record description of all data collected by GTF.

Figure 4 contains the SAS job used to select and analyze these records; Figure 5 contains a partial output.
// FIGURE 1. PROGRAM TO READ THE NCP COS
// JOB 0C70K NCP
// EXEC CRTC
// SMPLIST DD DSN=&&TEM7,DISP=(NEW,DELETE)
// CSIN DD DSN=&&TEM7,DISP=(NEW,DELETE)
// SAS PROGRAM TO READ AND ANALYZE A CDS OUTPUT FROM SMP
// DATA:
// OPTIONS TIME=90 LINESIZE=90 PAGESIZE=100
// LENGTH TYPE $8 STATUS $15 FMID $8.
// INPUT TYPE $8 STATUS $15 FMID $8.
// IF TYPE = 'SUP' THEN DO:
// OUTPUT
// RETURN
// END;
// IF TYPE = 'REQ' THEN DO:
// INPUT
// GO TO A
// END;
// DROP TTATUS, MMID, COUNT, DT, NYPE, LASTYPE;
// IF NYPE = 'STATUS' THEN DO:
// INPUT
// RETURN
// END;
// DROP TTATUS, MMID, COUNT, DT, NYPE, LASTYPE;
// IF TTATUS = 'DELETE' THEN DO:
// INPUT
// RETURN
// END;
// DROP TTATUS, MMID, COUNT, DT, NYPE, LASTYPE;
// IF TTATUS = 'START' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'MODIFY' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'RENAME' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'REMOVE' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'STOP' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'TYPE' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'USER' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'EXIT' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'JOB' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'ERROR' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NOPE' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE2' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE3' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE4' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE5' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE6' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE7' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE8' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE9' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE10' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE11' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE12' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE13' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE14' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE15' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE16' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE17' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE18' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE19' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE20' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE21' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE22' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE23' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE24' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE25' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE26' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE27' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE28' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE29' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE30' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE31' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE32' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE33' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE34' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE35' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE36' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE37' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE38' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE39' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE40' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE41' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE42' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE43' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE44' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE45' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE46' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE47' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE48' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE49' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE50' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE51' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE52' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE53' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE54' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE55' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE56' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE57' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE58' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE59' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE60' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE61' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE62' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE63' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE64' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE65' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE66' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE67' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE68' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE69' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE70' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE71' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE72' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE73' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE74' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE75' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE76' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE77' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE78' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE79' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE80' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE81' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE82' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE83' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE84' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE85' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE86' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE87' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE88' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE89' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE90' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE91' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE92' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE93' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE94' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE95' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE96' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE97' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE98' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE99' THEN DO:
// INPUT
// RETURN
// END;
// IF NYPE = 'NONE100' THEN DO:
// INPUT
// RET
FIGURE 3. TERMINAL DATABASE LISTING

NUCO ENGINEERING COMPUTER SERVICES TSO NETWORK TERMINALS THRU THE NCP AND LOCAL TERMINALS LISTED BY PHYSICAL UNIT (VTAM) ADDRESS

<table>
<thead>
<tr>
<th>VTAM_ID</th>
<th>CABLE</th>
<th>BUILDING</th>
<th>ADDRESS</th>
<th>SUBAREA</th>
<th>ROOM</th>
<th>MAKE</th>
<th>MAP_ID</th>
<th>SERIAL NO</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPRCT04</td>
<td>MP04</td>
<td>MP1</td>
<td>BIF</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>1-3278</td>
<td>27008</td>
</tr>
<tr>
<td>MPRCT05</td>
<td>MP05</td>
<td>MP1</td>
<td>BIF</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>1-3278</td>
<td>27011</td>
</tr>
<tr>
<td>MPRCT06</td>
<td>MP06</td>
<td>MP1</td>
<td>BIF</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>1-3278</td>
<td>27004</td>
</tr>
<tr>
<td>MRDAR01</td>
<td>MR01</td>
<td>NUSCO</td>
<td>BIF</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>1-3776</td>
<td>13566</td>
</tr>
<tr>
<td>MRDAR02</td>
<td>MR02</td>
<td>NUSCO</td>
<td>BIF</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>1-3278</td>
<td>9428</td>
</tr>
</tbody>
</table>

///# FIGURE 4. SVC 93 ANALYSIS PROGRAM
///# EXEC EXEC PC.M-AHPDRMP
///# /TAPE DD DSN=USER VTM3TF.DIS=CHR
///# SPACE=(CYL.100,10)
///# PRT  DD DSN=AHPRT.DISP=(,PASS)
///# SPACE=(CYL.10,ALSE)
///# DDN=(BLKSIZE=3901,RECLC=121,RECFM=FBA)
///# EXEC_Print DD DSN=VTAM,TAPE=C93
///# FIRST SELECT THE SVC 93 RECORDS FROM THE GIF FILE
///# DELETE
///# SAS PROGRAM TO READ AND ANALYZE THE RECORDS
///# EXEC EXEC NOPRINT,NDATE,LINESIZE=180.OBS+700,PAGESIZE=35'
///# IFILPO01 DD SYSOUT=*,CHARS=10,DISP=DELETE
///# DATA SVC93,INPUT=R1S REG15 REGIS REG18 ASCII
///# DROPT1ST R1S REGIS REGIS ASCII ; INPUT=R1S REGIS REGIS ASCII ;
///# IF TGET='SVC 991' THEN DO ;
///# INPUT=R1S USER_ID @CHARA. ;
///# R1S-12
///# DO WHILE (R1S NE 'R1S'); INPUT=R1S REGIS ASCII ;
///# IF R1S=23 THEN INPUT=R1S REGIS ASCII ;
///# ELSE DELETE ;
///# END;
///# TITLE SVC 93 ANALYSIS OUTPUT
///# TITLIE CONTENTS OF REGISTERS IN TPUT
///# TYPE="TGET"
///# FREQUENCY BAR CHART
///# AVG CONTENTS OF REGISTERS IN TPUT
///# TYPE="TGET"
///# LOCATION

##FIGURE 5. TPUT ANALYSIS OUTPUT
###AVG CONTENTS OF REGISTERS IN TPUT
####TYPE="TGET"
#####FREQUENCY BAR CHART

LENGTH MIDPOINT

DATA SVC93,
SHARE DD DSN=SYS,INPUT=R15 REGIS REGIS ASCII ;
DROPT1ST R1S REGIS REGIS ASCII ; INPUT=R1S REGIS REGIS ASCII ;
IF TGET='SVC 991' THEN DO ;
INPUT=R1S USER_ID @CHARA. ;
R1S-12
DO WHILE (R1S NE 'R1S'); INPUT=R1S REGIS ASCII ;
IF R1S=23 THEN INPUT=R1S REGIS ASCII ;
ELSE DELETE ;
END;
TITLE SVC 93 ANALYSIS OUTPUT
TITLE CONTENTS OF REGISTERS IN TPUT
PROC SORT ; BY TYPE ;
PROC MEANS MAXOEC=2;
TITLE SVC 93 ANALYSIS OUTPUT
TITLE CONTENTS OF REGISTERS IN TPUT
PROC CHART ; VBAR LENGTH
SYMBOL=* W=0000 MIDPOINT=5 TO 4100 BY 40 ;
PROBLEM: REBLOCK LINKLIBS

NUSCO recently migrated the data on all of its 3330 drives to 3350 devices. The IBM utilities, IEBGENER, IEMOVE, and IEBCOPY provide a vehicle to complete this process. Unfortunately, when copying a load module library, IEBCOPY does not reblock the library. Thus, a linklist library can be moved to a 3350 which contains maximum track size of 19069 bytes, but it will retain the block size of a 3330, 13030 bytes, wasting about one-third of each track. NUSCO knew of two ways to reblock: the first involved relinking all three thousand members of its load module libraries; the second involved using SAS with PROC PDSCOPY.

Obviously, relinking every member is an overwhelming task. SAS was by far the easiest solution.

Twelve batch runs were set up to copy the system libraries from the current SYSRES volume to an alternate resident volume. Each run consisted of the following three steps:

1. IEFBR14 - delete old library on alternate resident volume.
2. IEFBR14 - create new library on alternate resident volume.
3. SAS - PROC PDSCOPY to copy from resident volume to alternate resident volume.

Careful evaluation of the output of PDSCOPY is needed. Some libraries contain scatter load modules, which PDSCOPY does not handle. NUSCO found about thirty scatter modules in its linklist libraries. Each of these was linkedited with a batch job containing INCLUDE, ENTRY, and NAME control cards.

A second message to look for is "xxxx ALIAS not copied (member is not present)." For some reason, the original member has been deleted but the alias remains. Each of these cases must be carefully evaluated. NUSCO found two modules that had to be relinked as a result of this.

A third message to look for is "xxxxxx is not a load module." Member SYSCATLG in SYS1.NUCLEUS and ICECOMMA in SYS1.SORTLIB must be copied using IEBGENER.

SAS PROC PDSCOPY gives no warning when copying load modules with the "DC" attribute. It copies them without this attribute. Short record length load modules are found in the Network Control Programs that are loaded into the 3705. The NCP, PEP, and EP modules must be relinked. Each installation must be very careful with this property of PROC PDSCOPY.

SAS documentation warns that PL/1 modules cannot be copied with PDSCOPY. NUSCO used the PL/1 install procedure to relink these modules.

After copying all the system load module libraries on the resident system, the reblocked alternate system was loaded and ran with no problem. SAS PDSCOPY saved countless hours of system programmer time for this move. The total time for the move was eight manhours, roughly three wall clock hours of computer time, and about five minutes of 3033 CPU time to reblock all linklist and resident system load module libraries.

PROBLEM: UNCATALOGUED DUPLICATE DATASETS

SAS provides a very good method for reading the direct access volume table of contents (VTOC). The SAS.SAMPLE library contains a member called MAPDISK that will read the VTOC. NUSCO used this member to read the VTOC's of four volumes in the same batch run and produce a report of the two hundred largest datasets. Eight of the top twenty had the same data set name. After examining the problem, the systems programmers learned that the users were submitting batch jobs that created datasets with a disposition of (NEW,CATLG). The jobs would execute and catalog the dataset successfully the first time. On the second and succeeding runs, the datasets would be created but not catalogued. The poor user who forgot to change the disposition on the JCL would wonder why his job would execute normally with new data but his results remain the same.

Thus, NUSCO saw a need to eliminate the duplicate datasets. Since the users were accessing these datasets from TSO and the only dataset that the users knew of was the cataloged version, NUSCO decided to delete the uncatalogued duplicate DNAME. A minor change in the SAS logic could delete the oldest dataset and catalog the newest.

The NUSCO TSO catalog is an OS CVOL. The utility LISTCVOL is used to list the entries in this catalog. This report is written out to a scratch disk. SAS then reads this report and the VTOCs of the TSO packs. The lists of dataset names are sorted and merged; finally, the dataset names to be deleted are combined with JCL for IEFBR14 to create a job. This job is read into the internal reader and is placed on hold; it is released following a backup of the affected volumes. The systems and operations staff identify and notify those users who make these errors in an effort to avoid future mishaps of this kind. Figure 6 contains the SAS program for this operation. The prime reason for running this job is effective management of TSO DASD space. The growth of TSO datasets is a cancer that has to be continually treated; otherwise it spreads all over the system. The capabilities of SAS to read various types of data helps perform this task quickly and painlessly.
PROBLEM: VTAM SMS BUFFERS

VTAM, the Virtual Telecommunications Access Method, provides a tool for aiding in determining the number of buffers at startup time. These buffers are important to performance and operation of VTAM. VTAM virtually stops when it runs out of buffers. ACF/VTAM R2 does dynamic expansion and contraction of these buffers, but the expansion/contraction process consumes cycles. Overallocation of these buffers wastes the Common Systems Area (CSA), a limited resource. Storage Management Services (SMS) will write out the number of buffers used and allocated after a certain interval to the Generalized Trace Facility (GTF) dataset. The IBM trace print facility, AMDPRDMP, will print out all these records. The core to this problem is examining all of these records and determining a proper value for each of the buffers. NUSCO has a busy network which generates many SMS of print records. In order to derive a meaningful set of numbers, SAS reads in the print file from AMDPRDMP, selects only the SMS records, and then performs a quick analysis with PROC MEANS. A plot of buffer needs versus time aids in showing peak loads and duration of load. With this study, a valid count of each type of VTAM buffers can be determined.

To Obtain the SAS Analysis:
1. Activate the network.
2. Start GTF and collect USR records.
3. F VTAM,TRACE,TYPE=SMS,ID=VTAMBUF
4. Collect records for a day.
5. F VTAM,NOTRACE,TYPE=SMS,ID=VTAMBUF
6. Stop GTF.
7. Submit the SAS program.
8. Plug the 90% percent buffer values into ATCSTRxx and restart VTAM.

Figure 6 contains a listing of the SAS/GTF procedure; Figure 7 contains a partial output. SAS was thus very useful in solving this VTAM performance problem.

PROBLEM: NCP BUFFERS

As with many other MVS applications, the 3705 Network Control Program (NCP) will perform better with an intelligent choice for the size of its buffers. VTAM will write out the current buffer values and some other NCP tuning statistics to either the system console or to the SMF dataset. With SAS, it is an easy matter to read these records, accumulate some statistics, and analyze the results. NUSCO will study these numbers on occasion, after the addition of a new network node or a new application going online.

The ACF/VTAM Installation manual, SC27-0468-0, documents the tuning of the VTAM/NCP buffer values in Appendix C. Ratios of one buffer to another imply favorable or unfavorable performance for the NCP. The SAS program computes these ratios and allows the systems programmer to make his own judgement as to the goodness of his values. For instance, NUSCO found that its ratio of RDBUF to (MAXBFRU $\times$ CHRD) was too high. At the same time, the value of RDATN was high, indicating that its value of MAXBFRU was too low. This conflict must be examined on an overall basis; it would be difficult to have a SAS program predict the correct MAXBFRU values based on the input parameters. Thus, SAS is useful in helping to find current values, but it would be difficult to use to help predict a "good" value, as much judgment is needed in determining them. Figure 8 shows the SAS program to pick up these numbers from the SMF dataset, and Figure 9 is the result.

SUMMARY

SAS is a very useful tool for the systems programmer. In its initial applications, SAS is usually used to read and analyze the MVS SMF/RMF records. However, it is very easy to take advantage of the ability of SAS to read almost any dataset to solve other performance problems. Other features of SAS make it ideal for system maintenance and load module reblocking. There are countless other areas to employ SAS in; for example, this paper did not show any application of SAS in reading VSAM catalogs. By using some imagination, many problems can be solved with a "quick and dirty" SAS application.
** FIGURE 6. PROGRAM TO READ AMOPROMP SMS BUFFER TRACE RECORDS **

```plaintext
*0003582 JOB (VSP,008)
*MSGCLASS=A, NOTIFY=OLEKSIW, MSGLEVEL=1
*EXEC PGM=AMOPROMP
*TAPE DD DSN=USER.VTAMGSF, DISP=SHR
*SYSUT1 DD UNIT=SCRATCH, DISP=NEW, SPACE=(CYL,(1,1)), RLSE
*DCB=(LRECL=121, BLKSIZE=593, RECFC=FBK)
*SYSIN DD SYSOUT=*
*SYSIN DD *
*EDIT DDNAM=TAPE, USR=F00
END
```

** EXEC SABS9, OPTIONS='NODATE, LINESIZE=100, PAGESIZE=45'
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
/* FIGURE 8. PROGRAM TO READ SMF NCP */

DATE=TUNE.
DATA=TUNE(DROP=HOUR DATE),
PROC MEANS MAXDEC=2 BY NCP NAME ;
PROC PRINT UNIFORM ; BY NCP_NAME ;
TITLE1 FIGURE 9 ;
TITLE2 ACF/NCPO TO ACF/VTAM TUNING STATISTICS ;
TITLE3 NORTHEAST UTILITIES SERVICE COMPANY ;
TITLE2 RECORDS COLLECTED FROM SMF DATASETS ;
PROC MEANS MAXDEC=2 DATA=TUNE(DROP=HOUR DATE); BY NCP_NAME ;
PROC PLOT ;
%PILOT PLOT; PLOT IPIU=HOUR='2a'X/HOUR=1 TO 12 BY 1 ;

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>MEAN</th>
<th>STANDARD DEVIATION</th>
<th>MINIMUM VALUE</th>
<th>MAXIMUM VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_DLIR</td>
<td>50</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>CHRD</td>
<td>50</td>
<td>496.32</td>
<td>496.02</td>
<td>496.52</td>
</tr>
<tr>
<td>CHRD</td>
<td>50</td>
<td>491.36</td>
<td>480.90</td>
<td>497.80</td>
</tr>
<tr>
<td>ATTN</td>
<td>50</td>
<td>420.52</td>
<td>400.34</td>
<td>413.70</td>
</tr>
<tr>
<td>PIB4</td>
<td>50</td>
<td>671.34</td>
<td>738.32</td>
<td>663.00</td>
</tr>
<tr>
<td>PIB4</td>
<td>50</td>
<td>533.06</td>
<td>517.61</td>
<td>547.80</td>
</tr>
<tr>
<td>PIB4</td>
<td>50</td>
<td>481.52</td>
<td>465.13</td>
<td>492.00</td>
</tr>
<tr>
<td>PIB4</td>
<td>50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PIB4</td>
<td>46</td>
<td>0.93</td>
<td>0.07</td>
<td>1.00</td>
</tr>
<tr>
<td>PIB4</td>
<td>46</td>
<td>0.00</td>
<td>0.01</td>
<td>1.00</td>
</tr>
</tbody>
</table>

NORTHEAST UTILITIES SERVICE COMPANY

RECORDS COLLECTED FROM SMF DATASETS
NCP_NAME=NCPO03

511