Paper 281-27

SAS CLUSTER DISK MANAGEMENT:

USING SAS TO MANAGE VIRTUAL DISKS ON A CLUSTER OF FILE SERVERS

Haftan Eckholdt and Ezra Benun, DayTrends, Brooklyn, NY

ABSTRACT

Using SAS to manage low cost virtual drives on clusters of file servers is very simple and enables researchers to approach larger more complex problems. We will discuss methods for using SAS for: (1) managing very large virtual drives in file server clusters, (2) random and symmetric file naming, file seeking, file mapping, and management on these virtual drives, (3) visual and quantitative assessments of cluster usage and file distribution, and (4) provide crude versions of RAID backup redundancy. These methods allow research groups to acquire very large storage capacity at extremely low costs.

INTRODUCTION

While supercomputing in the form of parallel processing addresses the computational barriers to research, there are many other barriers preventing researchers for pursuing their ideas.

The challenges of data intensive projects include that production, acquisition, management, and analysis of data (files) distributed across many disks from many points in time. Writing, then finding, reading and reconciling these files often involves data loss rates beyond tolerance for tracking and analysis in longitudinal / repeated measurement projects. SAS MACRO® and the SAS X® command can be used like a robot to search drives and directories across networks in order to seek and map file structures for information regarding location, file characteristics, and variables. The product of file mapping can then be used to read eligible files for analysis.

Researchers today are challenged with managing rapidly growing multi site data structures, a trend that is likely to accelerate with internet and intranet evolution and usage. It is not unusual for analysts to manage and manipulate hundreds of thousands of files of varying formats residing in many known and unknown places.

A typical experiment or project often involves thousands of files, each with thousands of data points. It is not unusual for laboratories to collect a gigabyte of relevant data points on each subject or event. Nor is it unusual for the drive, directory (sub-sub-sub), and file naming schemes to have no symmetric logic, and no documentation. Some researchers have no idea where data have been stored and what they are named. This reality is not a sign of bad science, just bad technical management. And bad technical management means that files cannot be found for analysis. Lost data begins to look like bad science as many great hypotheses fall between the cracks of time and space along with their data. To date, there are no self-organizing software systems that can take the place of good technical management. So scientists are left to their own (de)vices.

The final straw for each of these barriers is cost. Our previous abstracts (W2129, W2930) describe low cost approaches supercomputing through parallel processing with clusters of workstations. This abstract describes another common barrier to high density data modeling: disk space. Our approach to this problem is to access disks with clusters of low cost file servers.

CHALLENGES / APPROACH

Hardware

There are several essential layers constructing a COW. A fileserver is needed to hold code, hold/distribute data, collect answers. A primary workstation is needed to write code, submit code (direct/ftp/etc), and collect files/answers. Many fileservers are needed to serve disks. Other issues to be discussed include sharing peripherals (monitors, keyboards, and mice), power, cooling, and communications. Readers should note that the approach to parallel COW processing discussed here is not very efficient in terms of network communications. Making clusters more efficient is discussed in great detail in Buyya (1999).

First of all, the fastest and cheapest file server is LINUX! Not only is Linux inexpensive, but it is easy to set up. And, most importantly, there is plenty of research to show that Linux servers are very fast. Once the number of clients exceeds about 8 workstations, there is no competition.

In this case, Red hat 6.2 [www.redhat.com] was downloaded and installed on a modified station (faster throughput with two hard drives [primary master + secondary master] set in RAID0 configuration with two Ethernet cards). You can purchase RedHat 7.0 with a beautiful new user interface, or you can download the very same thing for nothing. RedHat Linux has no limits on the number of attached clients. Buying support is (a) not very necessary, and (b) not very useful. This Samba Server, necessary to connect Microsoft NT workstations, was up and running in 12 minutes. Readers should note that I have faked this part of the COW in the past with a plug and play SNAP! Server which has no limits on the number of clients as well. For long standing COWS, the higher speed and life of a traditional fileserver is worth the additional cost.

Any number of file servers can be on the disk network. The configuration of each server was essentially identical, except for IP addresses and machine names. Actually one set of hard drives were set up and optimized, and the rest were cloned across the LAN with Symantec GHOST in 45 minutes! GHOST is part of *NORTON SYSTEM* WORKS, and a shareware version can get you most of the way there for nothing, although the full function version is worth the price. Note that each server had a market value of about $500 in September 2000 (500 Mhz, 128 Mb RAM, 80 Gb HDD, 100 mps Nic, SVGA).

Communication

Servers shared several keyboards/mice/monitors with Belkin KVM switches. Remember that none of the servers will be touched again after the first login in most cases. The network communication was handled as cheaply as possible using small hubs ($100 each). Long standing clusters should use more expensive and faster network switches ($1000-$2000 each).

A few cautions should be noted: (1) all server clusters need lots of smoothed electricity (many smaller UPS's are significantly cheaper and a single larger UPS); (2) all server slusters produce lots of heat, so there needs to be lots of cool breezes; and (3) in the event of a power outage, the UPS should be set to shut down all systems before the ambient temperature reaches critical limits… a few minutes in most rooms is a good rule of thumb, really; (4) all clusters eat hard drives for lunch, so use swap drawers, have plenty of GHOSTed clones on hand, and carefully monitor the warranty on any drive [hint: an IDE drive on a server will definitely burn up before its warranty expires].

Writing

Writing files is not very complicated. The underlying logic involves mapping many server hard drives to the analysis station, whereby each drive gets a SAS LIBNAME. The rest of the process follows a few simple rules.

Most workstations are limited in their number of mapped hard drives (A to Z) where A, B, C, D, and often E are used up locally. Thus there are about 20 letters remaining. This only limits the numbers of drives that can be simultaneously mapped, because is it simple to remap (in SAS) to other drives as the job demands change.

File naming can be used as a powerful source of information, although, code described later in this abstract will show that even random meaningless file naming can also be used.

Starting with the Fully Logical case, each file name represents as much information as possible in terms of location and content.

The second approach at the other extreme is Fully Random whereby the filenames have no relationship to the location or content of the file.

Management

The first step in seeking data files is to identify the hard drive(s) to be mapped. This is a relatively simple boot strap to the entire process, but it may require that the user "map" networked drives to the local workstation. The reader should note here that the user on the SAS processing workstation must have the privileges of a network administrator to do much of this work, and that the hard drives to be mapped have already be assigned share status, along with users and passwords. These procedures will not map non-local drives invasively without prior permissions and explicit privileges being established. The code below starts by turning off the X command feature with opens an Operating System window and waits for the user to "exit" the window before continuing to the next step. This is a critical start since hundreds of thousands of commands may be delivered to the local Operating System. Waiting for user input would turn an 8 hour job into an 8 month job. The following code then creates two SAS libraries (a temp library, and a long term library). This code then turns off the SAS log which will crash almost any operating system hard disk array with the files mapped and merged goes into the thousands. This code then cycles through a SAS MACRO® procedure that names drive N to be mapped.

```
OPTIONS PAGENO=1 NOXWAIT;

LIBNAME WORK    'C:\TEMP\SAS\';
LIBNAME SUGI    'C:\SUGI\2002\DATA\';

FILENAME NOLOG DUMMY;
PROC PRINTTO LOG=NOLOG;

%LET DRIVE = N;
```

The next step uses the SAS X® command which sends the quoted string to the operating system. This code simulates opening a local operating system window and typing

DIR N:\ > C:\SUGI\2002\DATA\DIR1.TXT

This text uses the Microsoft DOS "dir" command to list the contents of the N: drive root directory. The most important feature of this command is the ">" sign which tell the operating system to dump the output of the "dir" command to the following file rather than the screen. In this case, the target file is call DIR1.txt, and it will consist of ASCII text and reside in the SUGI library on the local C drive.

```
X "DIR N:\SUGI\DATA\ARCHIVE >
 C:\DIR1.TXT";
```

The output file looks something like the following:

```
Volume in drive N has no label.
 Volume Serial Number is 164A-18D6

 Directory of N:\

02/26/00  08:15a      <DIR>          FOUND.000
01/27/00  05:06p      <DIR>          I386
01/27/00  05:10p      <DIR>          SP3
01/27/00  05:23p      <DIR>          WINNT
01/27/00  05:21p      <DIR>          LANDRVR
06/28/00  01:46p       212,860,928 pagefile.sys
01/27/00  05:30p      <DIR>          Program Files
01/27/00  05:31p      <DIR>          TEMP
01/27/00  05:33p      <DIR>          WinUtils
01/27/00  05:34p      <DIR>          ToshUtil
01/27/00  05:34p      <DIR>          hddpwdnt
01/28/00  02:43p      <DIR>          Win32App
01/28/00  02:48p      <DIR>          SAS
01/28/00  03:21p      <DIR>          Multimedia Files
01/28/00  03:31p      <DIR>          lotus
01/28/00  04:18p      <DIR>          PASS60
02/18/00  07:45p      <DIR>          lomg_NT
06/30/00  12:48p                   0 DIR1.TXT
              18 File(s)    212,860,928 bytes
                             88,768,512 bytes free
```

As one can see, the N drive is someone's C: drive since it contains all of the Operating System and software directories that one would expect to find on the root. The following code will perform file mapping by reading the DIR output file as a data file. In this example, the SAS binary file will be called DIR1 and stored in the SUGI library. In Windows NT 4.0 workstation, the variables read in include file creation date, file creation time, including am/pm, the directory descriptor – whether or not the object is a file or directory, and finally, the file or directory name. None of these data appear until the 8th line, so the FIRSTOBS option must be used to skip over the file header garbage.

```
DATA SUGI.DIR1;
INFILE
"C:\DIR1.TXT" FIRSTOBS = 8;
INPUT
FDATE MMDDYY8.
FTIME TIME7.
AMPM $
FDIR $
FNAME $;

IF FDIR NE "<DIR>" THEN DELETE;

PROC SORT;
 BY FDIR;

DATA _NULL_;
 CALL SYMPUT ("SAMPLE",COUNT);
 STOP;
 SET SUGI.DIR1 NOBS=COUNT;

run;
```

Prior to using this code, which was written and optimized to run on a Microsoft Windows NT 4.0 workstation (SP3), the user must submit the quoted X command string and verify the structure of the DIR output file. The next lines in the code the perform the special task of acquiring the directories only, and creating a temporary SAS data file that only contains the directory tagged observations from the DIR output file. The number of directories in the DIR output file is also saved as a variable called sample to be used in the next set of MACRO procedures call Look.

This macro loops around once for each observation (directory) in

the DIR output file (ASCII version called DIR1.TXT and SAS version call DIR1.SD2) using the %DO loop until the %END at the bottom. For each directory on N:, a variable is created call SDIR which is space trimmed version of FNAME. This will allow the MACRO LOOK to then perform the same DIR X command on each and every directory in DIR1.SD2 which will be called DIR2.TXT and DIR2.SD2 respectively.

```
%MACRO LOOK;

%DO L = 1 %TO &SAMPLE;

DATA SUGI.DIR1;
 SET SUGI.DIR1;

IF _N_ = &L THEN
 CALL SYMPUT ("SDIR",TRIM(FNAME));

RUN;

X "DIR N:\&SDIR >
 C:\SUGI\2002\DATA\DIR2.TXT";

DATA SUGI.DIR2;
 INFILE
 "C:\SUGI\2002\DATA\DIR2.TXT" FIRSTOBS = 8;
 INPUT
FDATE2 MMDDYY8.
FTIME2 TIME7.
AMPM2 $
FDIR2 $
FNAME2 $;

DIR1 = "&SDIR";

IF FDIR2 NE "<DIR>" THEN DELETE;

PROC SORT;
 BY FNAME2;

DATA SUGI.DIR2;
 SET SUGI.DIR2;
 BY FNAME2;

 CALL SYMPUT ("SDIR2",PUT(FNAME2,Z8.));

RUN;
```

At this point, the reader should understand that this process can continue infinitely until there are not more subdirectories, with minor edits to the macro syntax and logic. A few additional procedures can then be used to bring about a single file structure that contains all of the files, and all of their directory tree information. Note that the first time around, a file must be created rather than appended (%* commented out below).

```
PROC TRANSPOSE
 DATA=SUGI.DIR2
 OUT=SUGI.DIR2H
 NAME=File
 PREFIX=FILE;
 VAR FNAME;

%*DATA SUGI.ARCHIVE;
data SUGI.dir2h;
 ATTRIB DIR1 LENGTH=$9;
 attrib DIR2 LENGTH=$8;
 SET SUGI.DIR2H;

DIR1 = "&DIR1";
DIR2 = "&DIR2";
PROC APPEND BASE=SUGI.ARCHIVE
 DATA=SUGI.DIR2H FORCE;

 %END;
%MEND LOOK;
%LOOK;
```

In this last step, a frequency distribution is established about the number of directories and files.

```
DATA SUGI.ARCHIVE;
 SET SUGI.ARCHIVE;

NFILE = 0;
IF FILE1 NE " " THEN NFILE = NFILE +1;
IF FILE2 NE " " THEN NFILE = NFILE +1;
IF FILE3 NE " " THEN NFILE = NFILE +1;
IF FILE4 NE " " THEN NFILE = NFILE +1;
IF FILE5 NE " " THEN NFILE = NFILE +1;
IF FILE6 NE " " THEN NFILE = NFILE +1;
IF FILE7 NE " " THEN NFILE = NFILE +1;
IF FILE8 NE " " THEN NFILE = NFILE +1;
IF FILE9 NE " " THEN NFILE = NFILE +1;
IF FILE10 NE " " THEN NFILE = NFILE +1;

PROC MEANS N MIN MAX MEAN SUM;
TITLE2 "ARCHIVE READING FOR SUGI 2002";
VAR NFILE;

PROC FREQ;
TABLES NFILE;

RUN;
```

A next step in file mapping might also involve a summary of all file names and file types. This will help in deciding which kinds of files will be acquired next.

```
%macro LOOK2;

 %DO A = 1 %TO 10;

  DATA FILE&A;
   SET SUGI.ARCHIVE;

  FILE = FILE&A;

  DOT = INDEX(FILE,".");

   FILEN = SUBSTR(FILE,1,DOT-1);
IF DOT > 0 THEN FILET = SUBSTR(FILE,DOT+1);

  KEEP FILE DIR1 DIR2 FILE FILEN FILET;

 %END;


 DATA SUGI.ARCHFILE;
  SET
 %DO B = 1 %TO 10;
  FILE&B
 %END;;

%MEND LOOK2;
%LOOK2;

PROC SORT;
 BY DIR1 DIR2 FILET;

PROC FREQ;
TABLES DIR1 DIR2;

PROC FREQ;
TABLES FILET;
 BY DIR1;

RUN;
```

Variable Mapping

It is also possible to determine and map the content of each file.

```
PROC CONTENTS OUT = %SCAN(&&DMS&A,&B);

DATA SUGI.ALLVARS;
   SET &DMS1 &DMS2 &DMS3;

* LIST ALL VARIABLES IN ALL DATASETS;
PROC CONTENTS;
TITLE2 'SUGI STUDY';

* PRODUCE TABLE OF FREQUENCY OF OCCURRENCE OF
EACH NUMERIC VARIABLE;
PROC FREQ;
TITLE3 'REDUNDANCY OF NUMERIC VARIABLES ACROSS
DATASETS';
TABLES NAME / LIST;
```

Productivity

It is also possible to assess disk usage and productivity. The following code will produce an HTML file with reporting statistics and a chart of time lags between files from common jobs or disks, etc.

```
X "DIR X:\REPORTA\TEST*.SAS7BDAT >
C:\COW&Z..TXT";

DATA COW&Z;
 INFILE "C:\COW&Z..TXT" FIRSTOBS = 6  LRECL=65
PAD;
 INPUT CREATED $ 1-8 CREATET $ 11-16 MIDDLE $
17-39 FILEN $ 40-64;

IF COMPBL(MIDDLE) = "<DIR>" THEN DELETE;
IF CREATED = " " THEN DELETE;

NAME =
COMPRESS(SUBSTR(FILEN,1,INDEX(FILEN,".")),".");
MACHINE = "DT0&Z";

RUN;

*X "NET USE L: /DELETE";
*RUN;

 %END;

DATA TEST.SPEED;
 SET
 %DO Y = 30 %TO 52;
  COW&Y
 %END;;

FILE = 1*SUBSTR(NAME,5,4);

PROC SORT;
 BY MACHINE FILE;

DATA TEST.SPEED;
 SET TEST.SPEED;
 BY MACHINE FILE;

DD = SUBSTR(CREATED,4,2) * 60 * 24;
HH = SUBSTR(CREATET,1,2) * 60;
MM = SUBSTR(CREATET,4,2) * 1;

IF SUBSTR(CREATET,6,1) = "p" and
SUBSTR(CREATET,1,2) lt 12 THEN HH = HH +
(60*12);
IF SUBSTR(CREATET,6,1) = "a" AND
SUBSTR(CREATET,1,2) = 12 THEN HH = HH - (60*12);
*IF (SUBSTR(CREATET,1,2) < 12 AND
SUBSTR(CREATET,6,1) = "a") then HH = HH +
(12*60);
*IF LAG1(HH) - HH > 2 THEN HH = HH + (12*60);

LENGTH = 1*(DD + HH + MM);

CALC = 0;
IF MACHINE = LAG1(MACHINE) THEN CALC = 1;

SPEED = LENGTH-LAG1(LENGTH);
IF CALC = 0 THEN SPEED = .;
IF SPEED GE 100 THEN SPEED = .;

**** html coding;

 data newspeed;
   set test.speed;
   length cowdrill $ 40;


   /* Assign targets for CITY values. */

 %DO X = 30 %TO 52;
   if machine="DT0&X" then
cowdrill="HREF='cow_reports.html#IDX&X'";
 %END;
 run;

 /* Define titles and footnotes. */
 title1 "Observed File Processing Speed (min)
for each coW";

 /* Define symbol. */
 symbol1 interpol=join
         value=dot
         height=3;

 /* Open ODS HTML body file. */
 ods html path=odsout
          body='cow_plots.html'
          nogtitle;

 /* Generate a plot of three variables  */
 /* that produces a legend.             */
 proc gplot data=newspeed;
    plot speed*file=machine /
         html=cowdrill
         html_legend=cowdrill;
 run;
 quit;

 /* Open the file for the PROC REPORT output. */
 ods html path=odsout
          body='cow_reports.html';

 /* Sort data set NEWTEMP in order by city. */
 proc sort data=newspeed;
    by machine file;
 run;
 quit;

 /* Clear the footnotes. */
 goptions reset=footnote;

 /* Suppress the default BY-line. */
 option nobyline;

 /* Print a report of monthly temperatures  */
 /* for each city.                          */
* title1 'File Processing Speed (min) for
Machine #byval(machine)';
```

```
* proc report data=newspeed nowindows;
*   by machine;
*   column machine file speed;
*   define machine / noprint group;
*   define file / display group;
*   define Speed / display group;

 proc means n min max mean data=newspeed;*
nowindows;
 title1 'File Processing Speed (min) for
Cluster';
  var speed;

 proc means n min max mean data=newspeed;*
nowindows;
 title1 'File Processing Speed (min) for Machine
#byval(machine)';
   by machine;
  var speed;

 run;

 ods html close;
 ods listing;
```

Redundancy

It is also possible to copy disks for a "poor mans RAID".

This code will copy one drive to another.

```
XCOPY Z: D: /E /V /F /H /R /S /K /Z
```

This batch file [call it MMDDNT.BAT] will create Date variables for DOS.

```
@ECHO OFF

                for /F "tokens=1-4 delims=/ "
%%i in ('date /t') do (
                   set DayOfWeek=%%i
                   set Month=%%j
                   set Day=%%k
                   set Year=%%l
                   set Date=%%i %%j/%%k/%%l
                )

                ECHO Date is %Date%
                ECHO Day of Week is
"%DayOfWeek%", Month is "%Month%"
                ECHO Day is "%Day%", Year is
"%Year%"
```

This batch file will copy specific file types.

```
CALL MMDDNT.BAT
MD C:\ARCHIVE\%Year%%Month%%Day%\sas
XCOPY X:\*.SAS C:\ARCHIVE\%Year%%Month%%Day%\sas
/E /V /S /EXCLUDE:7BDAT.TXT
```

CONCLUSION

Writing, Mapping, and Reading the data files for SAS jobs involving thousands of data files was previously impossible without very large main frame computers. Using a cluster of file servers, even very small and inexpensive ones, can give users access to several terabytes of disk space, with redundancy, at a fraction of the market cost. SAS, usually thought of as a data management and data analysis platform, can also be used to manage files, disks, and servers providing reasonable access, speed, and security with little additional cost or management overhead. For certain kinds of problems, SAS has the computational power to conduct the necessary analyses, but more importantly, the operational complexity to manage the problem on a networks of workstations and servers. Combining problem solving and data, file, network, and problem management in one software platform provides a critical cost savings for many researchers.

REFERENCES

Buyya, R. (1999). High Performance Cluster Computing: Architecture and Systems, volume 1. Prentice Hall: New Jersey.

Buyya, R. (1999b). High Performance Cluster Computing: Programming and applications, volume 2. Prentice Hall: New Jersey.

Foster, I. (1994). Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison_Wesley Publishing Company: New York.

Harris, Z. (1954). Distributional structure. Word. 10: 775-793.

Hill, T. (1998). WINDOWS NT: Shell Scripting. Macmillian Technical Publishing: Indianapolis, IN.

Singh, S. (1999). The Code Book. Doubleday: New York.

Ulam, S. (1976). Adventures of a Mathematician. Charles Scribner's Sons: New York.

Wilkinson, B. and Allen, M. (1999). Parallel Programming: techniques and applications using networked workstations and parallel computers. Prentice Hall: New Jersey.

CONTACT INFORMATION

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

Haftan Eckholdt
DayTrends, LLC
10 Jay Street, suite 504
Brooklyn, New York 11201
Work Phone: (917) 548-3178
Fax: (718) 522-3170
Email: haftan@thismind.com