Power Techniques for Processing Large Tape Data Sets Using the SAS® System in the MVS Environment

Michael A. Raithel, US Customs, Washington, DC

Abstract

Tape is one of the oldest and most dependable mediums for storing data sets in the MVS environment. SAS data libraries, OS data sets, and data base backup files are a few of the many types of data sets commonly stored on tape. Because tape is a relatively inexpensive media, it is particularly well suited for storing large data sets. But, the subsequent processing of large data sets stored on tape can be costly in terms of computer resources and elapsed time.

If you work in an organization that runs MVS, it is very likely that you will have to process large tape data sets sooner or later. Perhaps you already do.

This paper focuses on strategies for optimizing your processing of large tape data sets using the SAS System. It provides techniques that you can use to reduce computer overhead and elapsed time. Optimum methods for processing large SAS data libraries as well as OS data sets are covered in this paper.

Power Techniques for Processing All Tape Data Sets

Before discussing techniques parochial to SAS or OS tape data sets, we will examine those that pertain to both. Whether a tape data set contains a SAS data library or an OS data set, there are two factors that you should consider. First, you should pay special attention to the tape block size. The block size of a tape data set has a direct effect upon the number of EXCP's (I/O's) that will be needed to process it. The second consideration is for the type of tape device the data set is stored upon. "Square" tapes are better than "round" tapes, and the new, IBM® double density 3490-E tapes are better than the other square tapes. These two considerations are discussed in detail, below.

Power Technique #1: Maximize the Tape Block Size

The block size of a tape data set has an enormous effect upon the amount of computer resources needed to process it. The block size affects the total number of blocks that comprise a tape data set. Larger numbers of blocks require more EXCP's to be read from tape into computer memory. Larger numbers of blocks can also extend a data set across more tape volumes. Maximizing the block size of a tape data set minimizes the number of EXCP's required to process it and the number of volumes needed to store it.

Smaller block sizes allow fewer amounts of records to be stored per block in an OS data set. They allow fewer observations to be stored per page in a SAS data set. Thus, with smaller block sizes, more blocks are needed to store a tape data set. As an example, consider an OS tape data set with 100,000 records each 300 bytes long. If the block size of 6300 is used, it will require 4,762 blocks to store the entire data set. If the block size of 32,700 is used, it will require 918 blocks.

Always maximize the block size of tape data sets. *The maximum block size is 32,760 bytes.* You should block all of your tape data sets at or

near the block size of 32,760. Tape data sets created with the SAS System can be created at or near the maximum blocksize by using the following two SAS system options:

FILEBLKSIZE(*device-type*)=MAX
BLKSIZE(*device-type*)=MAX

The FILEBLKSIZE(*device-type*)= option above is invoked whenever an OS tape data set is created by a SAS program. "MAX" specifies that the blocksize is to be set at or near 32,760 bytes. By supplying the proper device-type in the option, all OS data sets created on that tape device type will have block sizes at or near 32,760 bytes.

The BLKSIZE(*device-type*)= option is used for SAS data sets. The value "MAX" sets all SAS data libraries created on tape to a block size of 32,760 bytes. It does not matter what block size the SAS data library had while on DASD; when it is copied to tape, it will have a block size of 32,760.

Here is an example of both in an OPTIONS statement:

```
options fileblksize(3480)=max
        fileblksize(3490)=max
        blksize(3480)=max
        blksize(3490)=max;
```

In the OPTIONS statement above, OS tape data sets created on IBM 3480 or 3490 tape devices will have a block size at or near 32,760 bytes. SAS data libraries created on IBM 3480 and 3490 tape devices will have a block size of 32,760 bytes.

The FILEBLKSIZE and BLKSIZE options can be specified in a variety of places in the SAS System. The best place for them to be specified is in the system configuration file. If they are not specified there, the next best place is in your user configuration file. Alternately, you can specify them in an OPTIONS statement in your autoexec file, or in your source programs. Ultimately, it does not matter exactly where you place them, as long as you do specify them somewhere!

Power Technique #2: Utilize the Latest Tape Technology

It is no secret that tape storage technology keeps getting better and better. "Better" means that greater numbers of blocks can be stored per volume on more modern device types. IBM 3420 tape devices can store about 160 Mbytes, while 3480's can store about 200 Mbytes. IBM 3490 tape devices can store about 400 Mbytes, and 3490E's can store about 800 Mbytes per cartridge.

If your organization has a mixture of tape device types, it is worthwhile for you to store your data sets on the most current tape device. By doing so, you can eliminate some tape mounts. Unless your organization has a robotic tape system, each tape mount requires a human intervention. This results in delays to your SAS batch jobs while operators locate tapes and mount them into tape units. The fewer tapes your data set occupies, the fewer resulting tape mount operations they will require when accessed. This translates into quicker batch job turnaround time.

For example, an OS tape data set with 2,426,922 records, each 600 bytes in length, blocked at 32,400 bytes spans 45 3480 tape volumes. The same data set spans seven 3490E tape volumes. To process the data set on 3480's requires forty seven tape mounts! That is enough to strain the patience of any human tape operator. And, think of the consequences if an abend occurs halfway through the job, forcing a rerun!

Power Techniques for Processing SAS Tape Data Sets

SAS data libraries stored on tape may be comprised of one or more SAS data sets. They

may span several, a dozen, or scores of tape volumes. MVS treats a SAS data library on tape as a single OS data set. SAS programmers know that within that "OS data set" (the SAS data library) the individual SAS data sets can be found. The power techniques described below pertain to SAS data libraries with multiple data sets that span one or more tape volumes.

Power Technique #3: Store CONTENTS as First Data Set on Tape

Have you ever used the CONTENTS procedure to map out a SAS data library stored on tape? If so, you may have noticed that it read every data block stored on every one of the data set's tape volumes to create the CONTENTS procedure listing. This happens because SAS sequential data libraries do not contain a library directory. Thus, the CONTENTS procedure must read the descriptor portion of every SAS data set on the tape to properly construct the listing.

Creating a CONTENTS listing can be an expensive operation, in terms of CPU time, EXCP's and wall clock time. The result of this costly expenditure of computer resources is not an analysis of data, but a map of the data sets and data elements stored in the data library. This may be absolutely necessary when you do not know exactly what is stored in the SAS tape data library. But, the fact remains that you must make a complete pass of the entire tape data set before you can actually access the data for analysis.

A common way to avoid this approach is to create a CONTENTS procedure listing of data libraries before they are copied to tape. These listings are often stored as hard copies with a manual procedure for associating them with the tape data sets. The administration and storage of such listings is often unmanageable in organizations with hundreds of SAS libraries stored on tape.

A more practical way to address this problem is to store a CONTENTS procedure output as the first SAS data set on the tape. The CONTENTS procedure data set contains one entry for each data element of each data set in the library. By dumping the CONTENTS procedure data set to DASD, you can determine the number and content of the data sets stored on the tape. Consequently, you incur only a single tape mount, and a limited number of EXCP's to garner complete CONTENTS information.

To produce a CONTENTS data set, do the following:

```
libname prodlib 'prod.data.set' disp=old;
proc contents data=prodlib._all_
     out=prodlib.aaconten;
run;
```

This code creates a SAS data set, named AACONTEN, that contains CONTENTS procedure information. The data set is stored in the PROD.DATA.SET data library. The CONTENTS data set should be created just before the data library is copied to tape via the COPY procedure. Since the COPY procedure copies data sets in alphabetical order of data set name, AACONTEN is the first data set on the tape.

When you implement this technique in your organization, use a standard name (such as "AACONTEN") for the CONTENTS data set. Make sure that this name is used by all SAS programmers that create SAS data libraries on tape. By doing this, all SAS programmers in your organization will know which data set they should dump to determine the contents of a particular SAS data library stored on tape.

To appreciate the savings this technique can offer consider a SAS data library with 37 data sets that spans 12 tape volumes. The CONTENTS procedure caused all 12 volumes to be mounted and consumed 9.31 seconds of CPU time and 75,633 EXCP's. Dumping the AACONTEN data set, caused a single tape

mount and used .11 CPU seconds and 161 EXCP's. Clearly, this is a technique that you want to make your own!

Power Technique #4: Use PROC COPY to Access Multiple SAS Data Sets on Tape

When you need to process multiple data sets from a SAS data library on tape, the best way to access them is via the COPY procedure. That is; use PROC COPY to first copy all of the data sets to DASD, and then process them with subsequent tasks within your SAS program.

The reason for this again lies in the fact that the SAS data library on tape does not have a directory. Thus, each DATA step that attempts to access a data set stored on tape causes the mounting of every tape that exists in the sequence before it. For instance, consider the SAS program, below, where data set CLIENTS is on tape #2 and data set PROSPECT is on tape #9 of a SAS data library that spans eleven tapes.

```
data clnts;
set tape1.clients;
...More SAS Code
run;

data prspct;
set tape1.prospect;
...More SAS Code
run;
```

In the example above, the first DATA step requires that 2 tapes be mounted. The second DATA step remounts the first two tapes, re-reads them, and reads subsequent tapes until it reaches the PROSPECT data set on tape #9. The only way for the SAS System to find a particular data set is to begin with the first tape and read through each block of each tape until it is encountered.

The SAS code below, makes a single pass of the first six tape volumes and dumps both data sets to DASD. Then subsequent DATA steps process the data.

```
proc copy in=tape1 out=work memtype=data;
     select clients prospect;
run;

...SAS DATA Steps
```

In the example above, the most important element is the MEMTYPE=DATA keyword. Without it, the COPY procedure reads every block of every tape of the SAS data library. Without MEMTYPE=DATA, the COPY procedure does not know that the program is actually looking for data sets. Consequently, it searches all of the tapes for any member type (DATA, VIEW, CATALOG, etc.) with the names of CLIENTS and PROSPECT. Always use the MEMTYPE= keyword!

The scope of the savings that can be realized by this technique can be illustrated in an example. Consider the 12 volume SAS tape data set mentioned in the Power Technique #3. In that library, the CLIENTS data set resides on tape #2 and the PROSPECT data set on tape #9. Using DATA steps to dump the data consumed 27.42 CPU seconds, 70,045 EXCP's and took 11 tape mounts. By contrast, the COPY procedure used 15.04 CPU seconds, 58,474 EXCP's and 9 tape mounts. Of course, if more than two data sets were to be processed, the CPU time, EXCP, and tape mount savings would be greater.

Power Techniques for Processing OS Tape Data Sets

Considering the volume of OS data sets stored on tape in most organizations, it is highly likely that you will need to process OS tape data sets sooner or later. You can easily utilize the SAS System to process OS data sets stored on tape. However, doing so in an efficient manner can be a challenge. This challenge is compounded when OS tape data sets span many volumes.

The power techniques described below can help you to reduce the computer resources expended when processing large OS tape data sets.

Power Technique #5: Influence the Sort Order of the Data Set

Try to influence the sort order of the records in OS tape data sets that you process. That is, act to ensure that the records are pre-sorted and then stored on tape in the order in which you will most frequently access them. If you are the person who creates the OS tape data sets, then you have direct control. If another group creates the data sets, then work with them to have the records put in the sequence that you need. In cases where several groups use the same tapes, determine the optimal sort sequence that can be used to meet everybody's needs.

Having the data pre-sorted helps you in two ways. First, you can use the SORTEDBY= DATA step option to assert a sort. This will save you from having to sort the data in your SAS program after you have read it from tape. Second, pre-sorting allows you to use Power Technique #6: Mapping the Sort Key Ranges to the Volsers. By knowing the sort fields and sort sequence, you can more easily map them out.

Here is how the SORTEDBY= data set option can be used:

```
filename tapefile 'prod.tape1' disp=shr;
libname  dasdfile 'prod.dasd1' disp=shr;

data dasdfile.custinfo(sortedby=lname);
infile tapefile;
input @1 lname $20.
    ...More SAS Statements
run;
```

In the example, above, the OS tape data set PROD.TAPE1 was pre-sorted into ascending sequence by the variable LNAME. When the DATA step executes, the SORTEDBY= option specifies that the observations in DASDFILE.CUSTINFO are sorted by order of the values stored in variable LNAME. No subsequent SAS System sort is necessary because, in this example, the programmer needed the observations stored in LNAME order.

Power Technique #6: Map Sort Key Range to Tape Volume Serial Numbers

Mapping sort key ranges to tape volume serial numbers can be a great help when you frequently access subsets of a large tape data set. To map out the sort key range, you need to run a simple SAS program that records the first and the last key value on each tape. An example of the simple SAS program is:

```
filename tape1 'prod.tape1' volume=B90210;

data _null_;
infile tape1 end=eof;

retain count;

input @1 lname $20.;
count + 1;

if _n_ = 1 then put 'first value= ' lname;
if eof then do;
   put 'last value= ' lname;
   put 'number of records= ' count;
end;
run;
```

The program above must be run once for each volume that comprises the tape data set. Thus, for each run, the VOLUME= keyword must be changed to specify the next tape volser in the OS tape data set sequence. After running the program above, you would create a map such as the one below:

Volser	First	Last	Count
B90210	Anderson	Bonham	2264579
B90125	Bowie	Bruford	2264575
B42755	Clapton	Daltry	2264580
B55274	Emerson	Frampton	2264570
B27554	Hayward	Lodge	2264578
B21000	Page	Plant	35000

Using the map above, you can access the records with the keys you want without having to read every tape that the data set spans. This saves tape mounts, CPU time and EXCP's because fewer tape data blocks are processed. For example, if you want to process records with the sort key sequence values of "BOWIE" through "DALTRY", you could code the following:

```
filename tape1 'prod.tape1'
   volume=(B90125,B42755);

data namefile(sortedby=lname);
infile tape1 end=eof;
input @1 lname $20.
       ...Other Input Variables
;
if lname > "DALTRY" then stop;
...Other SAS Program Statements
run;
```

In the example above, the VOLUME= keyword on the FILENAME statement is used. It specifies that only two of the tape volumes of data set PROD.TAPE1 are to be mounted. Tape B90125 is mounted first and tape B42755 is mounted after that. No other tape volumes are mounted. The immediate benefit in this example is that tape B90210 is not mounted, and does not have to be read through to get to the information on tapes B90125 and B42755. Hence, this Power Technique reduces tape mounts, CPU time, and EXCP's.

This power technique is only viable when subsets of information on the OS tape data set are frequently accessed. If the data set is always processed in its entirety then there is no need to map it. Similarly, if the tape data set is only used once or twice, it is probably not worth the overhead of running the SAS batch jobs that map out the key ranges.

Power Technique #7: Process Tape Data Sets in Multiple Concurrent Jobs

Sometimes it is inevitable that you are required to process data stored on every volume of a multi-volume tape data set. If you have done this before, then you may be aware of the potential pitfalls. Some common problems are:

- The greater the number of tape volumes, the longer the job runs
- If a tape I/O error is encountered you must rerun the entire job
- If an operator fails to mount any one of the tapes in a given time period, the job abends and must be rerun
- There may not be enough DASD space to hold a large output data set

These problems can be mitigated by running multiple concurrent SAS batch jobs. Each one of the batch jobs is coded to process several of the tapes in the OS data set. Each job reads the tapes, does some simple processing (such as record selection logic and data transformation), and dumps the data to SAS data sets on DASD. Further processing of the data is accomplished by concatenating the individual SAS data sets after they all exist on DASD.

As an example, consider the OS tape data set discussed in Power Technique #6, above. To process the tape entire data set, three separate batch jobs could be created and submitted concurrently. Each would process two of the tapes from the data set. The part of each program that reads the tape data set would look like this:

Program #1:

```
libname dasd1 'prod.dasd1';
filename tape1 'prod.tape1'
       volume=(B90210,B90125);

data dasd1.namefile(sortedby=lname);
infile tape1 end=eof;
input @1 lname $20.
```

```
        ...Other Input Variables
;
...Other SAS Program Statements
run;
```

Program #2:

```
libname  dasd2 'prod.dasd2';
filename tape1 'prod.tape1'
        volume=(B42755,B55274);

data dasd2.namefile(sortedby=lname);
infile tape1 end=eof;
input @1 lname $20.
        ...Other Input Variables
;
...Other SAS Program Statements
run;
```

Program #3:

```
libname  dasd3 'prod.dasd3';
filename tape1 'prod.tape1'
        volume=(B27554,B21000);

data dasd3.namefile(sortedby=lname);
infile tape1 end=eof;
input @1 lname $20.
        ...Other Input Variables
;
...Other SAS Program Statements
run;
```

After all three batch jobs have completed, it is a simple task to concatenate the data sets, now on DASD, to further process the data. Here is an example:

```
data bigincom;
set  dasd1.namefile
     dasd2.namefile
     dasd3.namefile;
by lname;
...Other SAS Program Statements
```

If you have the tape drives available in your organization, this technique can save you a considerable amount of time. As the number of volumes in an OS tape data set increases, so to does the viability of this technique.

Summary

It is probably inevitable that sooner or later you will have to process large data sets stored on tape. In doing so, you do not necessarily have to suffer long batch job run times and heavy computer resource utilization. The power techniques covered in this paper can dramatically reduce batch run time and computer resources. By judiciously applying them in your SAS programs, you can get to, and analyze your data more efficiently. And, that is what it is all about, isn't it?

Trademarks

References

Raithel, Michael A.
Tuning SAS Applications in the MVS Environment
Cary, NC:SAS Institute Inc.,1995
303pp.

Contact Information

Michael A. Raithel
PO Box 406
Garrett Park, Md. 20896
E-mail: 0006129710@mcimail.com