

Don't Fear the Network: Beat A Slow Network with Data Content Maps

Michael McCarthy PE, Austin Energy, Austin TX USA

ABSTRACT

While SAS programmers can improve benchmark speeds with better algorithms and faster computers, we cannot deliver results any faster than our network will allow. This can become a serious issue when we perform queries involving large data sets across several networked computers. Fortunately, there is a simple strategy to overcome this roadblock. A method to reduce the number of network data transactions is described, and a step-by-step example using BASE SAS® to create a Data Content Map is presented.

INTRODUCTION

SAS programmers frequently query SAS data tables located within their company's local Network Attached Storage (NAS). While the centralized location makes it possible for the SAS Management Console to control access, versioning, and script updates, the execution of SQL or PROC steps on this centralized data can involve transferring a large number records from the repository to the local SAS workspace. Thus, the speed of the local network can often create a data bottle-neck.

Since the SAS index can directly access a complete block of records from a table, rather than sequential pages of records, indexes provide insight into a possible solution. In particular, SAS reads in as many pages are necessary from disk into memory to locate all the records for a given index. When a SAS data set is queried using an index, the network transfer is negotiated just for the block of data pertaining to the index. Thus, the I/O is optimized for indexed records, while transferring n records any other way will depend upon the page size (BUFSIZE), page buffers (BUFNO), and network speed. Transferring records with an index should therefore be the fastest way to move data within the network.

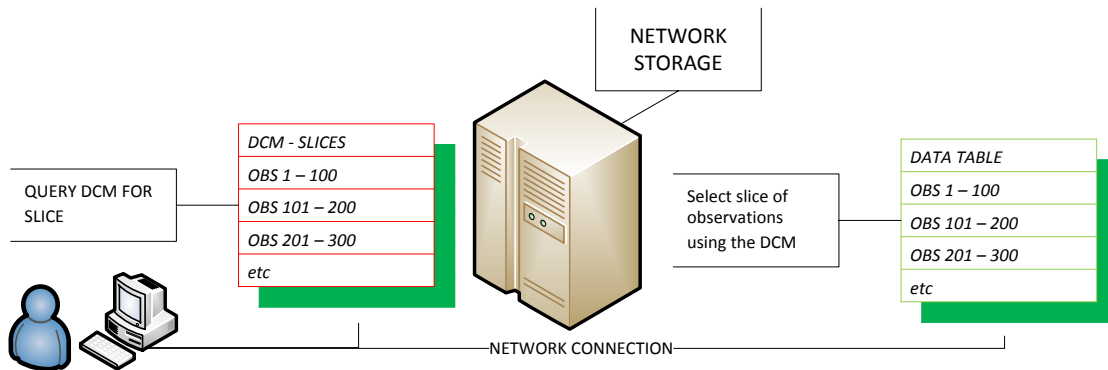
METHODOLOGY

To illustrate the value of transferring records in indexed blocks, we can utilize a Data Content Map (DCM). The DCM is essentially a Metadata table listing the starting and ending row for each value of the primary key within the data set. It's a potential alternative to extracting records with an index. Much like a SAS index, the DCM serves as a lookup table we can use to find a "slice" of records. In this way, the DCM reduces lag while transferring big data sets over a network.

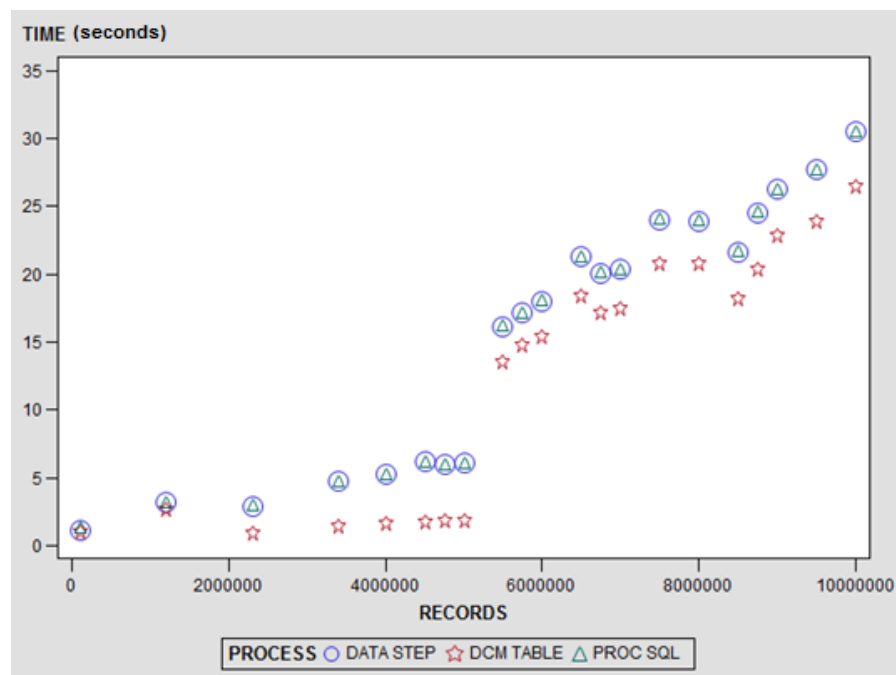
Record selection using the DCM is outline in the following steps:

- 1.) Determine the list of record keys from business report criteria.

- 2.) Join the first and last row to each key record using the DCM table.
- 3.) Select the records from the Source table using the beginning and ending observations you obtained from the DCM.



As we'd expect, the DCM approach is very comparable to transferring records by index. The chart below shows the time required to transfer a number of records for a single key value as the size of our indexed data set balloons. The "Data Step" and "PROC SQL" processes utilize the index, while the DCM table shows the times required to access the source table by observation. In Big O notation, all three processes follow $O(n)$, with stepwise changes in slope when record sizes exceed a network threshold (e.g. see 5×10^6 RECORDS in chart below).



To be sure, the SAS index will provide a more convenient method than a DCM for extracting specific records across the network. However, note that the DCM in the example table (1×10^9 records @100GB)

requires only 128 KB, while the SAS index file for the same source table requires 8 GB. That said, the I\O is still not as efficient as data access using an index. The table below shows how long it takes to transfer an entire data table from a networked location to the user workspace using each indexed value (INDEX TRANSFER), all the values from the DCM table (DCM TRANSFER), and the PROC COPY statement. In all cases, the SOURCE data table had a 66 KB page size:

INDEX TRANSFER	DCM TRANSFER	PROC COPY	SOURCE PAGE SIZE
28 min	32 min	55 min	66 KB

DATA CONTENT MAP BASE SAS® CODE

The BASE SAS® script shown in the **Appendix I** demonstrates how a content map is made. The sorted SOURCE table is read by KEY value, and the starting and ending observations for each of the KEY values are output to the DCM using the “firstobs” and “lastobs” data set descriptors. The BASE SAS® macro shown in **Appendix II** demonstrates how this can be used to filter the SOURCE data by beginning and ending observations. In step 2, the LIST of KEY values from a business query is joined to the DCM, thereby attaching the beginning and ending observations to each corresponding KEY. In steps 3-6, slices are selected using the range of observations with the macro FILTER, blocks are sent to the workspace, and the results are appended to the output data set BASE. Thus, only the required slices from the SOURCE data table are output to the workspace across the network.

ITERATIVE BENCHMARK IN BASE SAS®

The script used to benchmark data transfer times in this discussion is shown in **Appendix III**. The script iteratively creates sample data in a network location, sorts and indexes the data, and then benchmarks the time needed to extract specific records by primary key value. Here, the SAS user defines the number of script iterations and the number of records for each key (or slice size) in the Excel “Records.csv” table (see step 2). Then, macro DATATBL (see steps 11 - 17) creates table SOURCE with 100 KEY values (e.g. “KEY1”...“KEY100”) for each iteration and stores this table on the network. After the SOURCE table has been created, sorted, and indexed, a DCM file is then generated, and three benchmark macro scripts search the SOURCE table for “KEY50”. The total time (duration) required to search the SOURCE data table is determined using the “%sysfunc(datetime())” function, and results are appended to the output table TIME after each run.

INDEX TRANSFER BASE SAS® CODE

The most efficient I\O will occur when accessing data with an index. The script shown in **Appendix IV** demonstrates the transfer of any number of records via a LIST of indices. In this example, I’ve used the “INTO:” option to store the indices as a string of comma separated values in a macro variable called “%KEY_LIST” (see steps 4 and 5). The maximum size of this string is 32K characters (see REFERENCE 11-27).

CONCLUSION

The BASE SAS® code presented in this paper demonstrates the value of scrutinizing how records are transferred over your network. The most efficient way to transfer records is by index value, however, a Data Content Map (DCM) can provide a comparable alternative. Both strategies can be used to reduce the number of I/O operations and reduce the time required to deliver data to a SAS user.

PERMISSION TO PUBLISH

The authors of the paper/presentation have prepared these works in the scope of their employment with AUSTIN ENERGY and the copyrights to these works are held by AUSTIN ENERGY.

Therefore, AUSTIN ENERGY hereby grants to SCSUG Inc. a non-exclusive right in the copyright of the work to the SCSUG Inc. to publish the work in the Publication, in all media, effective if and when the work is accepted for publication by SCSUG Inc.

This the 7th day of September, 2017.

ACKNOWLEDGEMENTS

Special thanks to John Trowbridge, PE (DABI Engineering Supervisor, Austin Energy), Liz Jambor, DABI Manager, Austin Energy), and Debbie Kimberly (VP Customer Energy Solutions, Austin Energy).

REFERENCES

Paper 124-25 Power Indexing: A Guide to Using Indexes Effectively in Nashville Releases

Diane Olson, SAS Institute Inc., Cary NC

<http://www2.sas.com/proceedings/sugi25/25/dw/25p124.pdf>

Paper 255-2012 The Use and Abuse of the Program Data Vector

Jim Johnson, Ephicity Corporation, North Wales, PA

<http://support.sas.com/resources/papers/proceedings12/255-2012.pdf>

SAS® Blogs – SAS® timer – the key to writing efficient SAS code

<http://blogs.sas.com/content/sgf/2015/01/21/sas-timer-the-key-to-writing-efficient-sas-code/>

By Leonid Batkhan on [SAS Users](#) – Jan 2015

Paper 123-29 Creating and Exploiting SAS® Indexes

<http://www2.sas.com/proceedings/sugi29/123-29.pdf>

By Michael A. Raithel, Westat, Rockville, MD

Paper 11-27 Table Lookup: Techniques Beyond the Obvious

<http://www2.sas.com/proceedings/sugi27/p011-27.pdf>

By Nancy Croonen ir. Henri Theuwissen, Belgium

CONTACT INFORMATION

I value your comments. You can contact the author at:

Michael McCarthy, PE
Austin Energy, Customer Energy Solutions, Austin TX USA
michael.mccarthy@austinenenergy.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

APPENDIX I – DATA CONTENT MAP (DCM) SCRIPT

```
/******  
/* DCM SCRIPT  
/* WRITTEN BY: MMCCARTHY, PE  
/* DATE: 02/07/2017  
/* DESCRIPTION:  
/* CREATE A MAP OF RELATIVE POSITIONS FOR  
/* ALL RECORDS IN THE DATA SET BY KEY VALUE  
/******  
/*STEP*/  
/*1 */ libname SOURCE 'YOUR LIBRARY HERE';  
/*2 */ data SOURCE.DCM (DROP = obs);  
      set SOURCE.YOURTABLE;  
      by KEY;  
      Retain firstobs lastobs;  
      obs = _N_;  
/*3 */ if first.KEY then do;  
      firstobs=obs;  
      end;  
/*4 */ if last.KEY then do;  
      lastobs=obs;  
      output;  
      end;  
/*5 */ run;
```

APPENDIX II – FILTER BY DCM STORED PROCESS

```
/******  
/* FILTER BY DCM SCRIPT  
/* WRITTEN BY: MMCCARTHY, PE  
/* DATE: 02/07/2017  
/* DESCRIPTION:  
/* EXTRACT SLICE OF RECORDS FOR RANGE OF  
/* OBSERVATIONS. SELECT RECORDS BY KEY IN  
/* TABLE FOUND AND CREATE A LIST OF FIRST  
/* AND LAST OBS FOR EACH SELECTED RECORD  
/******  
/*STEP*/  
/*1 */ proc sql; /*CREATE A LIST OF RECORDS BY KEY FROM FOUND*/  
      create table LIST as  
      select distinct  
      a.KEY, b.firstobs, b.lastobs  
      from FOUND as a inner join SOURCE.DCM as b on a.KEY=b.KEY;  
      quit;  
/*2 */ %macro FILTER(START,STOP);  
      Data GETDATA;
```

```

        set SOURCE.YOURTABLE (firstobs=&START obs=&STOP);
run;
/*3 */ proc append base=BASE data=GETDATA FORCE;
        run;
/*4 */ %mend FILTER;
/*5 */ data _NULL_;
        set LIST;
        Call Execute ('%FILTER(START=' || firstobs || ',STOP=' || lastobs || ')');
run;
/*6 */ proc print data=BASE noobs;
        run;

```

APPENDIX III – ITERATIVE BENCHMARK SCRIPT

```

/*****
/* ITERATIVE BENCHMARK SCRIPT
/* WRITTEN BY: MMCCARTHY, PE
/* DATE: 08/30/2017
/* DESCRIPTION:
/* ITERATIVELY CREATE SAMPLE DATA AND SEARCH
/* OVER A COMPUTER NETWORK. CALCULATE THE
/* TOTAL TIME REQUIRED TO EXTRACT RECORDS
/* WITH A PARTICULAR KEY VALUE OR OBS RANGE
*****/
/*STEP*/
/*1 */ libname SOURCE 'YOUR NETWORK LIBRARY HERE'; /*DEFINE YOUR LIBRARY*/

/*2 */ PROC IMPORT OUT= WORK.RECORDS /*IMPORT RECORD LIST*/
        DATAFILE= "YOUR NUMBER OF KEY VALUES TO GENERATE.csv"
        DBMS=CSV REPLACE;
        GETNAMES=YES;
        DATAROW=2;
        RUN;
/*3 */ %macro TIMER(PROCESS,TIME,REC); /*TIMER MACRO – CREATE TABLE OF PROCESS TIMES*/
        Data NEWTIME;
        PROCESS      = "&PROCESS";
        TIME          = &TIME;
        RECORDS      = &REC;
run;
        proc append base=TIME data=NEWTIME FORCE;
run;
        %mend TIMER;

```

```

/*SCRIPT1 - DCM*/
/*4 */ %macro SCRIPT1(START,STOP);/*BUILD OUTPUT BY START|STOP OBS*/
    Data GETDATA;
    set SOURCE.BENCH (firstobs=&START obs=&STOP);
    run;
    proc append base=BASE data=GETDATA FORCE;
    run;
    %mend SCRIPT1;
/*SCRIPT1 - BENCHMARK*/
/*5 */ %macro BNMRK1(RECORDS);/*TIME THE DCM PROCESS*/
    %let _timer_start = %sysfunc(datetime());/* Start timer */
    data _NULL_;
    set LIST;
    Call Execute ('%SCRIPT1(START=' || firstobs || ',STOP=' || lastobs || ')');
    run;
    data _null_;/*RECORD DURATION*/
    format PROCESS $12.;
    dur = datetime() - &_timer_start;
    put 30*'-' / ' TOTAL DURATION:' dur time13.2 / 30*'-' ;
    PROCESS = "DCM TABLE";
    RECORDS = &RECORDS;
    /*RECORD TIME*/
    Call Execute ('%TIMER(PROCESS=' || PROCESS || ',TIME=' || DUR || ',REC=' || RECORDS || ')');
    run;
    %mend BNMRK1;
/*SCRIPT2 - DATA STEP*/
/*6 */ %macro SCRIPT2;
    Data SCRIPT2;
    set SOURCE.BENCH;
    where KEY in ("KEY50");
    run;
    %mend SCRIPT2;
/*SCRIPT2 - BENCHMARK*/
/*7 */ %macro BNMRK2(RECORDS);
    %let _timer_start = %sysfunc(datetime());/* Start timer */
    data _NULL_;
    Call Execute ('%SCRIPT2');
    run;
    data _null_;/*RECORD DURATION*/
    format PROCESS $12.;
    dur = datetime() - &_timer_start;

```



```

        put 30*'-' / ' TOTAL DURATION:' dur time13.2 / 30*'-' ;
        PROCESS = "DATA STEP";
        RECORDS = &RECORDS;
        /*RECORD TIME*/
        Call Execute ('%TIMER(PROCESS=' || PROCESS || ',TIME=' || DUR || ',REC=' || RECORDS || ')');
        run;
        %mend BNMRK2;
/*SCRIPT3 - SQL STEP*/
/*8 */ %macro SCRIPT3;
        proc sql;
        create table SCRIPT3 as
        select distinct
        a.*
        from a where a.KEY = "KEY50";
        quit;
        %mend SCRIPT3;
/*SCRIPT3 - BENCHMARK*/
/*9 */ %macro BNMRK3(RECORDS);
        %let _timer_start = %sysfunc(datetime());/* Start timer */
        data _NULL_;
        Call Execute ('%SCRIPT3');
        run;
        data _null_;/*RECORD DURATION*/
        format PROCESS $12.;
        dur = datetime() - &_timer_start;
        put 30*'-' / ' TOTAL DURATION:' dur time13.2 / 30*'-' ;
        PROCESS = "PROC SQL";
        RECORDS = &RECORDS;
        /*RECORD TIME*/
        Call Execute ('%TIMER(PROCESS=' || PROCESS || ',TIME=' || DUR || ',REC=' || RECORDS || ')');
        run;
/*10 */ %mend BNMRK3;

```

```

/*GENERATE THE BENCHMANRK DATA TABLE*/
/*11 */ %macro DATATBL(REC);
    Data SOURCE.BENCH (DROP = i k);
    format KEY $10.;
    array DATA (10) Value1 - Value10;
    DO i = 1 TO &REC;
        DO k = 1 TO 100;
            KEY = CATS("KEY",k);
            output;
        END;
    END;

    run;
/*SORT BENCH DATA SET*/
/*12 */ proc sort data=SOURCE.BENCH;
    by KEY;
    run;
/*INDEX BENCH DATA SET*/
/*13 */ data SOURCE.BENCH (index = (KEY)); /*CREATE AN INDEX ON KEY*/
    set SOURCE.BENCH;
    run;
/*CREATE THE DCM TABLE*/
/*14 */ data BENCH_DCM (DROP = obs);
    set SOURCE.BENCH;
    by KEY;
    Retain firstobs lastobs;
    obs = _N_;
    if first.KEY then do;
        firstobs=obs;
    end;
    if last.KEY then do;
        lastobs=obs;
    end;
    output;
end;

    run;
/*CREATE LIST- GET A LIST OF KEY VALUES*/
/*15 */ Data LIST;
    set BENCH_DCM;
    if KEY in ("KEY50") then output;
    run;

```

```

/*CREATE DATA TABLES ITERATIVELY AND BENCHMARK EACH PROCESS*/
/*16 */ Data _null_;
    RECORDS="&REC";
    Call Execute ('%BNMRK1(RECORDS=' || RECORDS || ')'); /*DCM BENCHMARK*/
    Call Execute ('%BNMRK2(RECORDS=' || RECORDS || ')'); /*DATA STEP BENCHMARK*/
    Call Execute ('%BNMRK3(RECORDS=' || RECORDS || ')'); /*SQL STEP BENCHMARK*/
    run;
/*17 */ %mend DATATBL;

/*CREATE AND BENCHMARK FOR DIFFERENT RECORD SLICES*/
/*18 */ Data _null_;
    set RECORDS;
    Call Execute ('%DATATBL(REC=' || REC || ')');
    run;

```

APPENDIX IV – INDEX TRANSFER BENCHMARK SCRIPT

```

/*****
/* INDEX TRANSFER BENCHMARK SCRIPT */
/* WRITTEN BY: MMCCARTHY, PE */
/* DATE: 08/30/2017 */
/* DESCRIPTION: */
/* USE A LIST OF INDEX VALUES TO TRANSFER A DATA */
/* TABLE FROM A NETWORKED LOCATION TO THE USER */
/* WORKSPACE */
*****/
/*STEP*/
/*1 */ libname SOURCE 'YOUR NETWORK LIBRARY HERE'; /*DEFINE YOUR LIBRARY*/
/*2 */ DATA LIST;
    SET SOURCE.SOURCE_DCM; /*GRAB ALL INDEXES FROM SOURCE*/
    run;
/*3 */ PROC SQL NOPRINT;
    SELECT DISTINCT KEY
/*4 */ INTO :KEY_LIST SEPARATED BY '"',"' /*USE THE INTO STATEMENT TO STORE LIST */
    FROM LIST ORDER BY KEY;
    QUIT;
/*5 */ %LET KEY_LIST = &KEY_LIST; /*STORE THE LIST IN MACRO VARIABLE KEY_LIST*/
/*6 */ %let _timer_start = %sysfunc(datetime()); /* Start timer */
/*7 */ DATA GETDATA;
    SET SOURCE.BENCH ; /*INDEXED BY KEY*/
    WHERE KEY IN ("&KEY_LIST");
    RUN;

```

```

/*6 */ data _null_; /*RECORD DURATION*/
format PROCESS $12.;
dur = datetime() - &_timer_start;
put 30*'-' / ' TOTAL DURATION:' dur time13.2 / 30*'-' ;
PROCESS = "COPY";
RECORDS = 10000000;
/*RECORD TIME*/
/*7 */ Call Execute ('%TIMER(PROCESS=' || PROCESS || ',TIME=' || DUR || ',REC=' || RECORDS || ')');
run;
/*8*/ %macro TIMER(PROCESS,TIME,REC); /*TIMER MACRO – CREATE TABLE OF PROCESS TIMES*/
Data NEWTIME;
PROCESS      = "&PROCESS";
TIME         = &TIME;
RECORDS      = &REC;
run;
proc append base=TIME data=NEWTIME FORCE;
run;
%mend TIMER;

```