Using SAS® system in very limited time/resource environment

Milorad Stojanovic, RTI International, RTP, North Carolina

Abstract

The paper will describe and comment on the use of the SAS system with significant constraints from two sides. First, a relatively short time was available for processing a relatively large amount of data. Second, the hardware was moderate both in speed and in capacity. The program used moderately complex macros, data _NULL_ steps, POINT data set option, and array processing. SAS data steps and Proc Sort were used for generating reports only. Processing in memory allowed the data to be handled quickly in comparison with the usual method of sorting and merging.

Introduction

The project had over 2.8 million of potentially affected records. The time for processing these records was restricted to 2 hours (real time) because other phases of the project had to be done and analysis data sets had to be generated for the client. My hardware was a PC with 233 MHz processor, 64 MB of RAM and 3GB hard drive. Windows 95 and SAS 6.12 were available on the PC. Data files resided on the server side behind a firewall. The project had 24 data files collected at 4 sites throughout the US. Data were collected once per week. Files were organized as fixed format ASCII files.

During the data collection phase it was found that some of the files contained records with inconsistent data values. In some cases (less than 1.5% of all cases) variables VAR6 or VAR7 had a value greater than VAR8. The client requested that all such records be removed from the affected data files. Of 24 data files, 14 files were affected. The problem was that just two of 14 files had those variables and all other files were linked via external keys. Also, the request was to preserve all input text files and output text files (after removal of affected records) in the same order and with the exactly same layout as they appeared at the input.

Four ways of solving the problem were considered

1. Transforming all affected text files to SAS data sets, sorting them, merging (removed inconsistent observations), and sorting them back to the initial sort order, and exporting such SAS data sets to text files.

2. Using SQL to do the job. It required using DTS (Data Transforming Services) to transfer data files or BCP (Bulk Copy Program) if you like to work from the command line. At that time the author did not have enough knowledge to achieve all those steps and timely delivery of first time data files for the client was critical.

3. Generate formatted values for the three key variables from the data set of selected cases and used in processing all 14 files. This idea was probably the best one, but because of lack of time for design and development idea, it was considered briefly and unfortunately was abandoned.

4. The fourth idea was to use the SAS system but to deal with text files as much as possible. Usage of the Data _NULL_ statement with Input and Output ASCII files was considered and, with the addition of a few performance improvements was accepted.

Steps in solution:

- FILE015 and FILE013 files were converted to SAS data sets.
- Combining FILE015 and FILE013 data sets produced DELETIND data set with the all cases with inconsistent data [(VAR6 > VAR8 +1) or (VAR7 > VAR8 +1)] . Each case in all 14 affected files was uniquely identified with VAR1, VAR2, and VAR3.
- Removed affected records from each of the 14 affected files. Macro %DELOBS processed records in the order as they were received file by file. An ARRAY statement was used to keep all values of said three variables and in the case of matching all three values, the record was removed. Immediately after a match was found a DO loop was exited. The RETURN statement prevented writing of that record to the output text file. It took advantage of processing data in RAM and reading of records sequentially. Macro variable &PRE and &SUF (prefix and suffix) allowed changes of INPUT to PUT statement and determined the physical destination of input and output files.
- Produced reports for Client.

Here is the main part of the program.

```
%macro DELOBS ( filnam, dat );
data REPORT (keep=tablXXX var2 var3 var4 ind)
      FILECNT (keep=tablXXX var2 ) ;
  length tablXXX $ 8;
  array arr(200,3) 3 arr1 – arr600 ;
  retain tablXXX '        ' ind  arr ;
tablXXX = "&filnam" ;
```

```
%let pre=IN ; /* Input */
%let suf=1 ;
%&dat ;
  if _N_ = 1 then do;
       do RecID = 1 to maxN;
         set DeletInd Nobs = maxN point=RecID;
         arr(RecID,1) = var11 ;
         arr(RecID,2) = var21 ;
         arr(RecID,3) = var31 ;
       end;
  end;
  do RecID = 1 to maxN;
    if arr(RecID,1) = var1 and
       arr(RecID,2) = var2 and
       arr(RecID,3) = var3 then do;
         ind = 1;
         output report;
         return;
    end;
  end;
%let pre=; /* Output */
%let suf=;
%&dat;
run;

proc freq data=FILECNT;
  tables siteid*tablXXX / noprint
       out=FILESUM ( keep=var2 tablXXX count
                          rename=(count=var3tot));
run;

proc append base = rlib.FILESUM data=FILESUM;
run;

proc datasets;
  delete FILECNT;
run;
proc append base = rlib.REPORTA data=REPORT;
run;
%mend DELOBS;

* Application of DELOBS on all affected files;
% DELOBS ( FILE001, fl01 )
% DELOBS ( FILE002, fl02 )
% DELOBS ( FILE003, fl03 )
% DELOBS ( FILE004, fl04 )
% DELOBS ( FILE005, fl05 )
% DELOBS ( FILE006, fl06 )
% DELOBS ( FILE007, fl07 )
% DELOBS ( FILE008, fl08 )
% DELOBS ( FILE009, fl09 )
% DELOBS ( FILE010, fl10 )
% DELOBS ( FILE011, fl11 )
% DELOBS ( FILE012, fl12 )
% DELOBS ( FILE013, fl13 )
% DELOBS ( FILE014, fl14 )
```

The author produced some quantitative measures about the data processing described above. The records were from 96 characters to 963 characters in length, and between 38,000 and 2,200,000 records were transferred (read/write). The estimated minimum number of transferred characters each week was over 2.3 GB in less than 2 hours (real time).

Conclusion

In solving this problem I did not want to apply so called “brute force”, by which I mean using a more powerful PC (with faster microprocessor, more RAM, bigger and faster hard drive). I tried to solve the problem with the available resources. The advantage of using array processing over merging data sets was that processing data in RAM is 3 to 4 levels of magnitude faster than accessing data on the hard drive. The usual number of inconsistent cases in the DELETIND SAS data set was between 40 and 60. This means in average the program needed to check 20 – 30 array cells to get the answer if case was removed and all 40 to 60 array cells if the case was kept.

Further Research

One more improvement can be the use of formatted values for three key variables instead of an array sequential search. Binary search, which is standard for formatted values, would drop the number of array searches from 60 (max number) to a maximum of 6 searches (under assumption that the number of observations in DELETIND is 63 or less). I believe it could improve performance.

Acknowledgements

Thanks to Daniel Pratt, Mani Medarametla, and Laura Burns from RTI International for their suggestions while preparing this paper.

Author Contact Information

Milorad Stojanovic
RTI International
Research Computing Division
800 Park
RTP, NC 27709
(919) 541-7376 milorad@rti.org