SAS® GLOBAL FORUM 2020

MARCH 29 – APRIL 1
WASHINGTON, DC

USERS PROGRAM

Programmatic Tools for Parallel Processing in SAS®

TAP TO GO BACK TO KIOSK MENU

Leah Durbak
The Urban Institute

Please use the headings above to navigate through the different sections of the poster

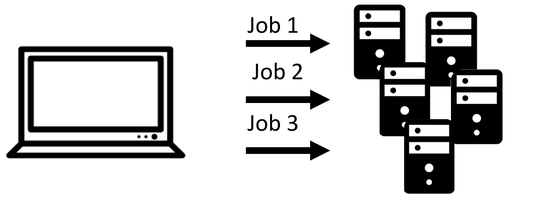
Summary

This poster features three tools for efficiently leveraging parallel processing using the SAS Grid within the Chronic Conditions Warehouse Virtual Research Data Center (CCW VRDC):

- the SAS macro facility,
- programs that write programs, and
- Automated log check programs.

What is parallel processing in the SAS® Grid?

- Multiple programs running simultaneously, each "job" using dedicated resources
- Supports big data processing
- Parallel processing occurs via workload distribution across a grid cluster of computers

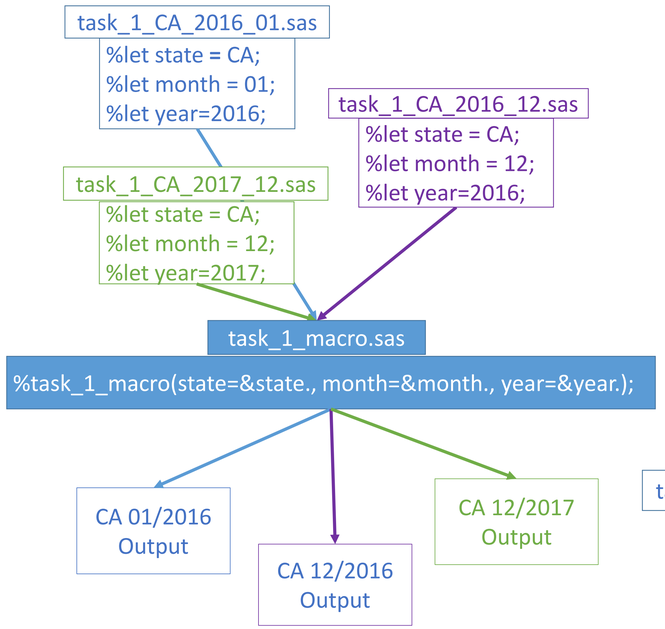


This poster uses examples based on work in the Chronic Conditions Warehouse:

- Source for Centers for Medicare and Medicaid (CMS) research data
- Accessed via the Virtual Research Data Center (VRDC)
- CCW VRDC contains 322+ billion records
- 115+ million beneficiaries
- Uses SAS EG and the SAS Grid environment for data processing

Tool 1: SAS Macros

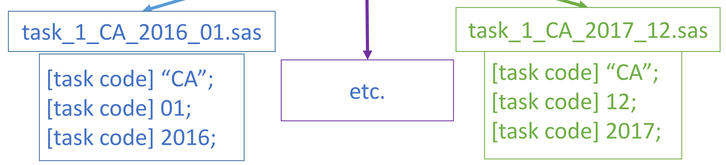
- Parameterize working code to be run over and over with difference parameter values



Tool 2: Write Programs with a Program

- Use loops to generate .sas programs with macros that resolve parameters into the body of the programs

```
%let state = CA;
%macro loop();
    %do month = 1 %to 12;
        %do year = 2016 %to 2017;
        file "file_path/task_1_&state._&year._&month..sas";
        put ' [task code] " ' "&state." '" ;';
        put ' [task code] ' "&month." ';';
        put ' [task code] ' "&year." ';';
        %end;
    %end;
%mend;
```



Tool 3: Automated Log Checks

- Programs that read through an entire directly of logs and extract pertinent information for speed-checking logs

Summary of Log Files
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014 folder

File	n	Description
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014/02_OT_CA_pullheaders_2014_01.log	475	WARNING:
	3	values have been converted
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014/02_OT_CA_pullheaders_2014_02.log	475	WARNING:
	3	values have been converted
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014/02_OT_CA_pullheaders_2014_03.log	475	WARNING:
	3	values have been converted

Summary of Batch Log Durations

filenm	max	min	duration
02_OT_OH_pullheaders_2014_01_20190908_221959_sasgsub.log	09SEP19:00:46:10	08SEP19:22:20:29	2:25:41
02_OT_OH_pullheaders_2014_02_20190908_221959_sasgsub.log	09SEP19:00:02:51	08SEP19:22:20:37	1:42:14
02_OT_OH_pullheaders_2014_03_20190908_221959_sasgsub.log	09SEP19:00:08:18	08SEP19:22:20:32	1:47:46
02_OT_OH_pullheaders_2014_04_20190908_221959_sasgsub.log	09SEP19:00:37:21	08SEP19:22:20:24	2:16:57
02_OT_OH_pullheaders_2014_05_20190908_221959_sasgsub.log	08SEP19:23:26:58	08SEP19:22:20:39	1:06:19
02_OT_OH_pullheaders_2014_06_20190908_221959_sasgsub.log	09SEP19:00:20:36	08SEP19:22:20:35	2:00:01
02_OT_OH_pullheaders_2014_07_20190908_221959_sasgsub.log	09SEP19:01:28:44	08SEP19:22:20:29	3:08:15
02_OT_OH_pullheaders_2014_08_20190908_221959_sasgsub.log	09SEP19:01:28:00	08SEP19:22:20:22	3:07:38

Programmatic Tools for Parallel Processing in SAS®

Leah Durbak
The Urban Institute

TAP TO GO BACK TO KIOSK MENU

Please use the headings above to navigate through the different sections of the poster

Summary

This poster features three tools for efficiently leveraging parallel processing using the SAS Grid within the Chronic Conditions Warehouse Virtual Research Data Center (CCW VRDC):

- the SAS macro facility,
- programs that write programs, and
- Automated log check programs.

Data tables for examples: finder_file and bene_key

```
/*The finder file is for a single state with a year and
month element that indicates for which years and months we
want the eligibilty data for each bene*/
data finder_file;
   input bene_link_key $ state $ year month;
   datalines;
a12345 AL 2014 01
b78912 AL 2014 01
c98751 AL 2014 01
c98751 AL 2014 02
d23467 AL 2014 02
;
run;
/*This is the table that crosswalks between the
bene_link_key and the bene_id for all states*/
data bene_key;
   input bene_link_key $ state $ bene_id;
   datalines;
a12345 AL 456
b78912 AL 789
c98751 AL 254
d23467 AL 896
f67753 OH 345
g89754 OH 876
h78427 WY 543
;
run;
```

Data tables for examples: elig_2014_01 and elig_2014_02

```
/*eligibility tables for 01/2014 and 02/2014, which includes data
for all states*/
data elig_2014_01;
   input bene_id state $ eligibility $;
   datalines;
456 AL .
789 AL MDCD
254 AL MDCD
896 AL MDCD
345 OH MDCD
876 OH .
543 WY MDCD
;
run;

data elig_2014_02;
   input bene_id  state $ eligibility $;
   datalines;
456 AL MDCD
789 AL MDCD
254 AL MDCD
896 AL MDCD
345 OH MDCD
876 OH MDCD
543 WY MDCD
;
run;
```

Programmatic Tools for Parallel Processing in SAS®

Leah Durbak
The Urban Institute

TAP TO GO BACK TO KIOSK MENU

Please use the headings above to navigate through the different sections of the poster

Summary

This poster features three tools for efficiently leveraging parallel processing using the SAS Grid within the Chronic Conditions Warehouse Virtual Research Data Center (CCW VRDC):

- the SAS macro facility,
- programs that write programs, and
- Automated log check programs.

Tool 1: SAS Macros

- Parameterize working code to be run over and over with difference parameter values
- Provides consistent output given identical input data and identical programming needs
- Reduces typographical errors of reusing code
- Must manually create/save separate programs with different parameter values in order to be batch submitted to the SAS Grid using the %bsas() macro available in the CCW

Example:

- The macro %join_ff_to_elig() in the **join_ff_to_elig_macro.sas** program can be called from separate programs as seen here – **ff_to_elig_AL_2014_01.sas** and **ff_to_elig_AL_2014_02.sas** .
- Programs must be created/saved manually to be batch submitted to the SAS Grid using the %bsas() macro available in the CCW.
- The batch submission macro, %bsas(), is available to CCW users and documentation for this macro can be found on the secured CCWData.org website.

Program: join_ff_to_elig_macro.sas

```
%macro join_ff_to_elig(year, month, state);
       proc sql;
/*this macro function creates a table for each input state, year, and month
that contains the eligibility data for that state's finder file beneficiaries
for that year and month using the bene_link_key table to crosswalk between
the finder file and the monthly eligibility table, and prints out that data*/
       create table elig_&state._&year._&month. as
       select ff.state, ff.year, ff.month,
                     bk.bene_id,
                     el.eligibility
              from finder_file ff left join bene_key bk
              on ff.bene_link_key = bk.bene_link_key and ff.state=bk.state
              left join elig_&year._&month. el
              on bk.bene_id=el.bene_id
              where ff.state="&state." and
       ff.year=&year. and
       ff.month=&month.;

              title "Eligibility data for &state., &month./&year.";
              select * from elig_&state._&year._&month.;
              title;
       quit;
%mend;
```

Program: ff_to_elig_AL_2014_01.sas

```
%join_ff_to_elig(year=2014,month=01,state=AL);
```

Program: ff_to_elig_AL_2014_02.sas

```
%join_ff_to_elig(year=2014,month=02,state=AL);
```

Program: batch_submit.sas

```
%bsas(dua_555555/lmd/ff_to_elig_AL_2014_01.sas);
%bsas(dua_555555/lmd/ff_to_elig_AL_2014_02.sas);
```

Programmatic Tools for Parallel Processing in SAS®

Leah Durbak
The Urban Institute

TAP TO GO BACK TO KIOSK MENU

Please use the headings above to navigate through the different sections of the poster

Summary

This poster features three tools for efficiently leveraging parallel processing using the SAS Grid within the Chronic Conditions Warehouse Virtual Research Data Center (CCW VRDC):

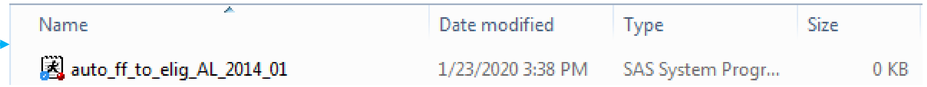
- the SAS macro facility,
- programs that write programs, and
- Automated log check programs.

Tool 2: Write Programs with a Program

- Again, parameterize working code so that it will work for whatever set of parameters you want (in this example, state and month/year)
- A DATA _NULL_ step can be used to create a .sas file
- Use PUT statements to write parameterized code in the DATA _NULL_ step
  - Use a spreadsheet program to easier format text for PUT statements
- Place the DATA _NULL_ step into a loop that will resolve the specified parameters
- The loops generate all the separate programs you need with specified parameters resolved
- Troubleshooting and adjusting code is easier because the program text is contained within all the programs.

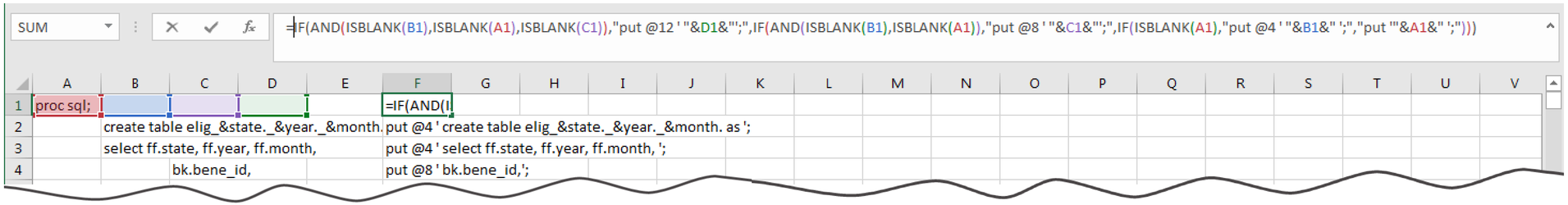
Create a .sas program from a SAS program

```
/*update this file path to your appropriate CCW directory*/
%let myfilepath = D:\Users\LDurbak\Box Sync\My Box Notes\My stuff\Blog;
data _null_;
     file "&myfilepath./auto_ff_to_elig_AL_2014_01.sas";
run;
```



- Every line that will be “printed” to our automatically-generated programs must be within a PUT statement. I set up an Excel worksheet with a text concatenation formula in Column F that will partially* format SAS code pasted in Columns A-D into PUT statements
  - *Partially: you need to manually format parameters that will be resolves and “” vs ‘’ in the code

Inset PUT statements generated from Excel formatting into DATA _NULL_ step



Continued on next slide

Programmatic Tools for Parallel Processing in SAS®

Leah Durbak
The Urban Institute

TAP TO GO BACK TO KIOSK MENU

Please use the headings above to navigate through the different sections of the poster

Summary

This poster features three tools for efficiently leveraging parallel processing using the SAS Grid within the Chronic Conditions Warehouse Virtual Research Data Center (CCW VRDC):

- the SAS macro facility,
- programs that write programs, and
- Automated log check programs.

Tool 2: Write Programs with a Program

- Paste the formatted PUT statements from Excel into the DATA _NULL_ statement
- Adjust “” and ‘’ around the parameters that should be resolved in the program text and quotations

Inset PUT statements generated from Excel formatting into DATA _NULL_ step

```
%let state=AL;
%let year=2014;
%let month=01;
data _null_;
    file "&myfilepath./auto_ff_to_elig_AL_2014_01.sas";
    put 'proc sql; ';
    put @4 'create table '"elig_&state._&year._&month." 'as ';
    put @4 'select ff.state, ff.year, ff.month, ';
    put @8 'bk.bene_id,';
    put @8 'el.eligibility';
    put @4 'from finder_file ff left join bene_key bk ';
    put @4 'on ff.bene_link_key = bk.bene_link_key and ff.state=bk.state
';
    put @4 'left join elig_&year._&month. el ';
    put @4 'on bk.bene_id=el.bene_id ';
    put @4 'where ff.state= "'"&state."'" and  ';
    put @8 'ff.year='"&year."'and ';
    put @8 'ff.month='"&month."';';
    put @12 '';
    put @4 'title "Eligibility data for '"&state."',
  '"&month."'/'"&year."'"; ';
    put @4 'select * from '"elig_&state._&year._&month."'; ';
    put @4 'title; ';
    put 'quit; ';
run;
```

Parameters will be resolved inside “”, but not ‘’

Format quotation marks

Code resulting from previous DATA _NULL_ step

```
proc sql;
    create table elig_AL_2014_01 as
    select ff.state, ff.year, ff.month,
        bk.bene_id,
        el.eligibility
    from finder_file ff left join bene_key bk
    on ff.bene_link_key = bk.bene_link_key and ff.state=bk.state
    left join elig_&year._&month. el
    on bk.bene_id=el.bene_id
    where ff.state= "AL" and
        ff.year=2014 and
        ff.month=01 ;

    title "Eligibility data for AL, 01/2014";
    select * from elig_AL_2014_01;
    title;
quit;
```

Continued on next slide

Programmatic Tools for Parallel Processing in SAS®

TAP TO GO BACK TO KIOSK MENU

Leah Durbak
The Urban Institute

Please use the headings above to navigate through the different sections of the poster

Summary

This poster features three tools for efficiently leveraging parallel processing using the SAS Grid within the Chronic Conditions Warehouse Virtual Research Data Center (CCW VRDC):

- the SAS macro facility,
- programs that write programs, and
- Automated log check programs.

Tool 2: Write Programs with a Program

- Add a macro function with loops to fill in the specified parameter values

Putting loops around the working DATA _NULL_ step to automatically create month-year programs

```
%let state=AL;
%macro loop_years_months;
%do loop_yr = 2014 %to 2018;
     %let year=&loop_yr.;
     %do loop_mnth=1 %to 12;
     %let month=%sysfunc(putn(&loop_mnth.,z2.)); /*code within %sysfunc() maintains leading zeros in single-digit months*/
     data _null_;
          file "&myfilepath./auto_ff_to_elig_AL_2014_01.sas";
          put 'proc sql; ';
          put @4 'create table ' "elig_&state._&year._&month." ' as ';
          put @4 'select ff.state, ff.year, ff.month, ';
          put @8 'bk.bene_id,';
          put @8 'el.eligibility';
          put @4 'from finder_file ff left join bene_key bk ';
          put @4 'on ff.bene_link_key = bk.bene_link_key and ff.state=bk.state ';
          put @4 'left join elig_&year._&month. el ';
          put @4 'on bk.bene_id=el.bene_id ';
          put @4 'where ff.state= "'"&state."'" and  ';
          put @8 'ff.year=' "&year."' and ';
          put @8 'ff.month=' "&month."' ;';
          put @12 ' ';
          put @4 'title "Eligibility data for '"&state."', '"&month."'/'"&year."'"; ';
          put @4 'select * from '"elig_&state._&year._&month."'; ';
          put @4 'title; ';
          put 'quit; ';
     run;
     %end;
%end;
%mend;
%loop_years_months;
```

Directory listing of the newly-created month-year programs

Name	Date modified	Type	Size
auto_ff_to_elig_AL_2014_01	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_02	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_03	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_04	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_05	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_06	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_07	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_08	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_09	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_10	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_11	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2014_12	1/23/2020 4:26 PM	SAS System Progr...	1 KB
auto_ff_to_elig_AL_2015_01	1/23/2020 4:26 PM	SAS System Progr...	1 KB

Programmatic Tools for Parallel Processing in SAS®

TAP TO GO BACK TO KIOSK MENU

Leah Durbak
The Urban Institute

Please use the headings above to navigate through the different sections of the poster

Summary

This poster features three tools for efficiently leveraging parallel processing using the SAS Grid within the Chronic Conditions Warehouse Virtual Research Data Center (CCW VRDC):

- the SAS macro facility,
- programs that write programs, and
- Automated log check programs.

Tool 3: Automated Log Checks

Batch submitting SAS programs in the SAS Grid results in two types of logs:

- usual .log files, which contain notes, errors and other important messages
- sasgsub.log batch files that lists periodic status checks on server activity for the submitted program

Several published articles include code to that will extract important messages from multiple .log files at once. I personally use the %scanlog() macro function created by Amit Baid and published in PharmaSUG (Baid, 2009). This function pulls out pertinent messages from a .log file, such as WARNING and ERROR, to alert you when a closer inspection of a .log file is created:

https://www.lexjansen.com/pharmasug/2009/po/PO25.pdf

Paper AD20

Catch the Bad Guys!!!
A Utility Program to Check SAS® Log Files
Amit Baid, ICON Clinical Research, Smyrna, GA

ABSTRACT

SAS log files are generated whenever a SAS program is run. Checking the log file is an important part of the validation process and enables us to make our programs error free. Usually we look for messages such as "ERROR" or "WARNING", but in this paper I am going to discuss some other messages which we also do not want to see in our log file. The utility program presented in this paper scans through one or more SAS log files in a specific folder for all the potential error messages. It provides a summary report which will assist a programmer in fixing the code to get rid of those messages. Going through this summary report is better than going through each SAS log file and finding all the unwanted messages. This program was developed on a PC SAS environment and works with both 8.x and 9.x versions of SAS.

HTML output of %scanlog() macro

Summary of Log Files
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014 folder

File	n	Description
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014/02_OT_CA_pullheaders_2014_01.log	475	WARNING:
	3	values have been converted
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014/02_OT_CA_pullheaders_2014_02.log	475	WARNING:
	3	values have been converted
/sas/vrdc/users/ldu880/files/dua_024376/UI/lmd/CA/OT_CA/pullheaders_2014/02_OT_CA_pullheaders_2014_03.log	475	WARNING:
	3	values have been converted

Extracting timings from batch log files can also be a helpful tool to identify bottlenecks in processing and document program durations for time estimation purposes. Modeled from the %scanlog() macro function, I created a similar macro %gettimings() that extracts the difference between the first and last status check in any sasgsub.log file in a single directory to provide a listing of timings.

HTML output of %gettimings() macro

Summary of Batch Log Durations

filenm	max	min	duration
02_OT_OH_pullheaders_2014_01_20190908_221959_sasgsub.log	09SEP19:00:46:10	08SEP19:22:20:29	2:25:41
02_OT_OH_pullheaders_2014_02_20190908_221959_sasgsub.log	09SEP19:00:02:51	08SEP19:22:20:37	1:42:14
02_OT_OH_pullheaders_2014_03_20190908_221959_sasgsub.log	09SEP19:00:08:18	08SEP19:22:20:32	1:47:46
02_OT_OH_pullheaders_2014_04_20190908_221959_sasgsub.log	09SEP19:00:37:21	08SEP19:22:20:24	2:16:57
02_OT_OH_pullheaders_2014_05_20190908_221959_sasgsub.log	08SEP19:23:26:58	08SEP19:22:20:39	1:06:19
02_OT_OH_pullheaders_2014_06_20190908_221959_sasgsub.log	09SEP19:00:20:36	08SEP19:22:20:35	2:00:01
02_OT_OH_pullheaders_2014_07_20190908_221959_sasgsub.log	09SEP19:01:28:44	08SEP19:22:20:29	3:08:15
02_OT_OH_pullheaders_2014_08_20190908_221959_sasgsub.log	09SEP19:01:28:00	08SEP19:22:20:22	3:07:38