SAS® Grid Job Submission and Monitoring from the SAS® Information Delivery Portal

Adolfo Lopez
Valence Health, Chicago, IL
245-2013

Introduction

You can use this section to introduce your poster. The implementation of SAS® Grid Computing presented several challenges including providing end users the ability to batch submit SAS® jobs to the grid directly from their workstations. It is important to note that Valence Health uses terminal servers to provide most staff access to SAS® applications, which means that no SAS® applications are installed on users desktops. In order for users to submit jobs to the grid they must log on to one of the terminal servers and either submit the job interactively via SAS® Enterprise Guide or in batch mode using the SAS® Grid Manager Client Utility. This of course assumes that the code file is stored on a network location that is accessible from the terminal server. In order to streamline the user submission process we developed this method, which utilizes SAS® Stored Processes and the Information Delivery Portal, to batch submit code files stored locally on the user's desktop to the SAS® Grid for processing and monitor the processing of the job on the grid. This paper assumes that you have a basic understanding of Hyper Text Markup Language (HTML), SAS® Stored Processes and the SAS® Information Delivery Portal.

SAS® INFORMATION DELIVERY PORTAL

The SAS® Information Delivery Portal is part of the SAS® Business Intelligence Platform which provides a Web-based user interface that provides users the ability to navigate and access a wide variety of information from both SAS® and third party sources. The information delivered via the portal includes reports, charts, Web applications, documents, and links to internal or external Web pages using portlets to organize information. The Stored Process Portlet enables the display of Stored Process output in static HTML format.

SAS® STORED PROCESSES

SAS® provides the ability to store programs on the server which can then be executed by an application such as the Information Delivery Portal. Stored processes can be used for a variety purposes and can access any data source or file that is available to the SAS®. Since Stored Processes are stored on the server and are administered by the SAS Metadata Server they can be called by client applications and can be modified from one central location using various applications.

JOB SUBMISSION STORED PROCESS PORTLET

The initial Stored Process, Grid_Submit_Program_Upload (Appendix A), displays an interactive HTML form, Figure 1, that allows the end user to browse their local file system for a SAS® program file which is uploaded to the grid and submitted for processing. The Stored Process consists of a series HTML statements embedded in a Data Step whose output is displayed in the Stored Process Portlet. Once the program file is submitted, a second Stored Process, Grid_Submit_Job (Appendix B), is triggered.

The second Stored Process, Grid_Submit_Job, completes the submission of the selected code file in a series of steps. The first step checks the users directory on the grid for the existence of the programs directory, if the directory does not exist it is created otherwise the Stored Process continues to the next step. The second step outputs the contents of the submitted program file to the users programs directory using the same file name as the original submitted file. The third step submits the contents of the program file that was created in the users programs directory and calls the SASGSUB command (See Grid Computing in SAS® 9.3, Second Edition) to submit the program file to the grid for processing.

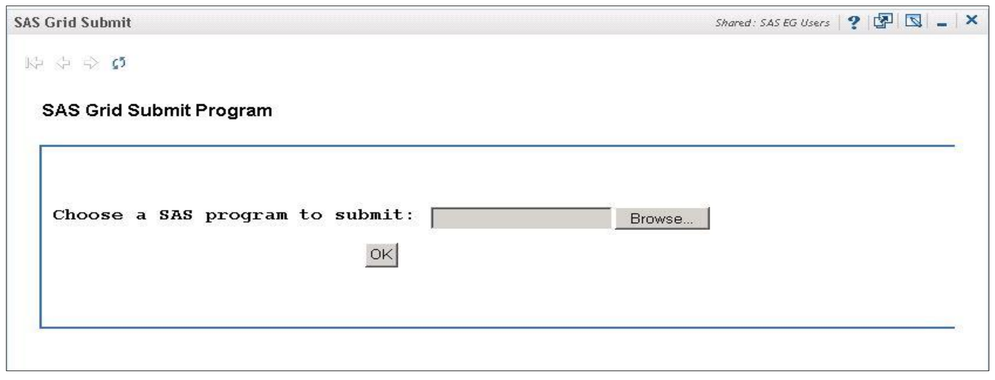
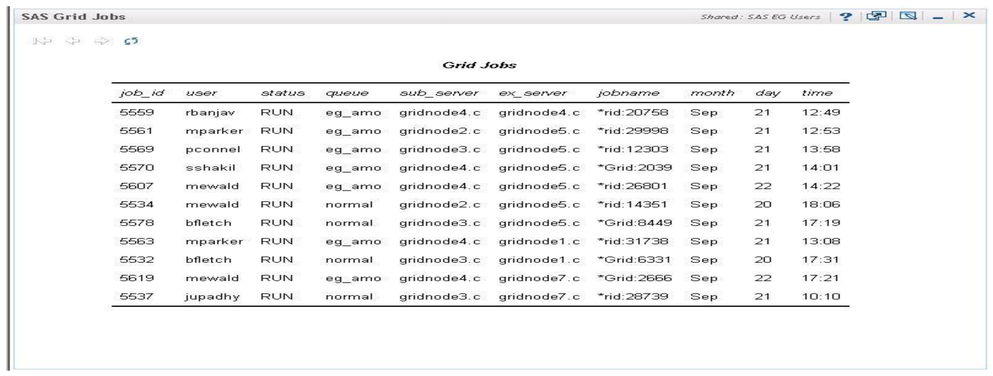


Figure 1.

JOB STATUS STORED PROCESS PORTLET

The Job Status Stored Process, *GSUB_JobStatus (Appendix C)*, Portlet allows users to monitor the status of jobs that have been submitted to the grid for processing. Using the Platform Suite for SAS® the Stored Process submits the Platform Load Sharing Facility (LSF) "bjobs" command to obtain the status of jobs that have been submitted to the grid. The output from the command is piped into a file which is used in a data step to load the information into a data set. The Stored Process then uses the Print Procedure in order to display the contents of the data set in the Stored Process Portlet.



job_id	user	status	queue	sub_server	ex_server	jobname	month	day	time
5559	rbanjav	RUN	eg_amo	gridnode4.c	gridnode4.c	*rid:20758	Sep	21	12:49
5561	mparker	RUN	eg_amo	gridnode2.c	gridnode5.c	*rid:29998	Sep	21	12:53
5569	pconnel	RUN	eg_amo	gridnode3.c	gridnode5.c	*rid:12303	Sep	21	13:58
5570	sshakil	RUN	eg_amo	gridnode4.c	gridnode5.c	*Grid:2039	Sep	21	14:01
5607	mewald	RUN	eg_amo	gridnode4.c	gridnode5.c	*rid:26801	Sep	22	14:22
5534	mewald	RUN	normal	gridnode2.c	gridnode5.c	*rid:14351	Sep	20	18:06
5578	bfletch	RUN	normal	gridnode3.c	gridnode5.c	*Grid:8449	Sep	21	17:19
5563	mparker	RUN	eg_amo	gridnode4.c	gridnode1.c	*rid:31738	Sep	21	13:08
5532	bfletch	RUN	normal	gridnode3.c	gridnode1.c	*Grid:6331	Sep	20	17:31
5619	mewald	RUN	eg_amo	gridnode4.c	gridnode7.c	*Grid:2666	Sep	22	17:21
5537	jupadhy	RUN	normal	gridnode3.c	gridnode7.c	*rid:28739	Sep	21	10:10

Figure 2.

Conclusion

Utilizing Stored Processes the capabilities of the SAS® Information Delivery Portal can extended beyond the standard uses that have been developed as part of the products core capabilities. No longer is the Information Delivery Portal limited to just information delivery but it can also be utilized as an interactive tool to gather user input. That input can then be utilized in a variety of ways including dispatching user selected jobs.

Reference

Anesta, Thomas. "SAS Fifth Dimension Tips and Techniques." SAS Fifth Dimension Tips and Techniques. N.p., n.d. Web. 24 May 2012. <http://sas-tips.com/sas-tips/sheetsunix/mkdirs.shtml>.

SAS Institute Inc. 2012. Grid Computing in SAS® 9.3, Second Edition. Cary, NC: SAS Institute Inc.

SAS Institute Inc 2011. SAS® 9.3 Intelligence Platform: Web Application Administration Guide, Second Edition. Cary, NC: SAS Institute Inc.

About Valence Health

Based in Chicago, Valence is a privately-held company that serves clients nationwide and is committed to strong client partnerships. Our multidisciplinary staff has experience in the clinical, management, operational, and financial challenges of health care organizations – and can efficiently integrate people and processes to improve quality and sustain success. Our core capabilities offer flexibility and scalability in meeting targeted goals or in creating integrated solutions that address complex challenges.

SPECIAL THANKS TO

Vic Andruskevitch, Valence Health
David Glemaker, SAS Institute, Inc.

A copy of this paper along with all the associated code may be obtained by scanning the QR code.

Appendix A - Grid_Submit_Program_Upload

```
*ProcessBody;

ods html file='/sasprod/users/alopez/pgms/stp_code/html/temp.htm' style=Styles.NormalPrinter;

%STPBEGIN;
Title Justify=Left "SAS Grid Submit Program";
data _null_;
file print;

put '<html>';
put '<form action="http://vbi.valencehealth.com/SASStoredProcess/do1" method="post" enctype="multipart/form-data">';
put '<input type="hidden" name="_program" value="/Projects/Stored_Processes/Grid_Submit_Job">';
put '<table border="0" cellpadding="5">';
put '   <tr>';
put '      <th>Choose a SAS program to submit:</th>';
put '      <td><input type="file" name="myfile" device="files" accept="text/*"></td>';
put '   </tr>';
put '   <tr>';
put '      <td colspan="2" align="center"><input type="submit" value="OK"></td>';
put '   </tr>';
put '</table>';
put '</form>';
put '</html>';
run;
%STPEND;
ods html close;
```

Appendix B - Grid_Submit_Job

```
*ProcessBody;

%global _ODSSTYLE;
%let _ODSSTYLE=Journal;

%STPBEGIN;
options mprint mlogic symbolgen source;
%let usr = %sysget(USER);
%macro mkdirs(name)                                                    ;
%put %str(---------->name exists %sysfunc(fileexist(&name)));
%if %sysfunc(fileexist(&name))=0 %then %do                    ;
data _nulls_                                                          ;
    num_nodes=0                                                  ;
    line="&name"                                                 ;
    do ii=1 to length(line)                                      ;
      if substr(line,ii,1)='/' then num_nodes=sum(num_nodes,1) ;
      if substr(line,ii,1)='/' then pos=ii                   ;
    end                                                          ;
    word1=scan(line,1,'/')                                       ;
    word2=scan(line,2,'/')                                       ;
    word3=scan(line,3,'/')                                       ;
    word4=scan(line,4,'/')                                       ;
    word5=scan(line,5,'/')                                       ;
    word6=scan(line,6,'/')                                       ;
    word7=scan(line,7,'/')                                       ;
    word8=scan(line,8,'/')                                       ;
    word9=scan(line,9,'/')                                       ;
    call symput('word1',substr(word1,1,length(word1)))       ;
    call symput('word2',substr(word2,1,length(word2)))       ;
    call symput('word3',substr(word3,1,length(word3)))       ;
    call symput('word4',substr(word4,1,length(word4)))       ;
    call symput('word5',substr(word5,1,length(word5)))       ;
    call symput('word6',substr(word6,1,length(word6)))       ;
    call symput('word7',substr(word7,1,length(word7)))       ;
    call symput('word8',substr(word8,1,length(word8)))       ;
    call symput('word9',substr(word9,1,length(word9)))       ;
    call symput('num_nodes',compress(num_nodes))          ;
run                                                              ;
    %put %str(---------->num_nodes=&num_nodes)              ;
    %do ii=1 %to &num_nodes                                  ;
      %put %str(---------->word&ii=&&word&ii)            ;
    %end                                                         ;
    let node=                                                    ;
    %do ii=1 %to &num_nodes                                 ;
      %let node=&node/&&word&ii                                  ;
      %if %sysfunc(fileexist(&node))=1 %then %do           ;
        %*put %str(---------->node=&node exists)         ;
      %end                                                       ;
      %else %do                                                  ;
        %put %str(---------->node=&node does not exist)  ;
        %if &ii=1 %then %do                                      ;
          %put %str(---------->first node must exists);
          %goto exit_mac                                     ;
        %end                                                     ;
      %end                                                       ;
    %end                                                         ;
    %let node=                                                   ;
    %do ii=1 %to &num_nodes                                 ;
      %let node=&node/&&word&ii                                  ;
      %if %sysfunc(fileexist(&node))=1 %then %do           ;
        %put %str(---------->node=&node exists)          ;
      %end                                                       ;
      %else %do                                                  ;
        %let jj=%eval(&ii-1)                                     ;
        %put %str(---------->jj=&jj)                       ;
        %let previous=                                           ;
        %do ss=1 %to &jj                                         ;
          %let previous=&previous/&&word&ss                ;
        %end                                                     ;
        %put %str(---------->previous=&previous)         ;
        %put %str(---------->node=&node created)         ;
 x "cd &previous"                                                ;
 x "mkdir &node"                                                 ;
      %end                                                       ;
   %end                                                          ;
%exit_mac:
%end                                                             ;
%else %do                                                        ;
  %put %str(---------->new directory exist &name)           ;
%end                                                             ;
%mend mkdirs                                                     ;
%mkdirs(/sasprod/users/&usr/pgms);


proc printto log="/sasprod/users/&usr/pgms/stp_gsub.log" new;
run;

%put &_WEBIN_FILENAME;
%let filenm = &_WEBIN_FILENAME;

%let pgmfile = /sasprod/users/&usr/pgms/&FILENM;

filename outfile "/sasprod/users/&usr/pgms/&FILENM";

data _null_;
     length data $1;
     infile &_WEBIN_FILEREF recfm=n;
     file outfile recfm=n;
     input data $char1. @@;
     put data $char1. @@';
  run;

%macro mygsub(mypgm);
  data _null_;
  command="&gsconfigdir./sasgsub -gridsubmitpgm &&&mypgm";
  call system(command);
  run;
%mend mygsub;
%mygsub(pgmfile);

data a;
Status = "&FILENM has been submitted for processing on the grid.";
run;

proc print data=a noobs;
run;

%STPEND;
```

Appendix C - GSUB_JobStatus

```
*ProcessBody;

%global _ODSSTYLE;
%let _ODSSTYLE=Journal;

%STPBEGIN;

%macro gjobs_allusers;
filename jobs pipe "bjobs -u all -a";

   data jobs;
   infile jobs firstobs=2 dlm=" " missover;
   length job_id $20.  user $20. status $20. queue $20. sub_server $20.
        ex_server $20. jobname $20. month $20. day $20. time $20.;
   input  job_id $ user $ status $ queue $ sub_server $ ex_server $
        jobname $ month $ day $ time $;
   run;

title "Grid Jobs";
proc print data=jobs noobs;
run;title;

%mend gjobs_allusers;

%gjobs_allusers;

%STPEND;
```