Paper 3144-2019

Process Manager Batch Job Monitoring

Pradeep Ponnusamy, Royal Bank of Scotland

ABSTRACT

The paper explains how one can best use the information stored in a history.log file that contains information about the jobs running in process manager. A SAS® program can be written to read the history.log file to identify failed jobs, long-running jobs, CPU consumption of the jobs, execution time of jobs, and so on.

INTRODUCTION

Being a SAS® administrator or working in a team which supports SAS jobs running in a production environment, wouldn't it be good to get notified about batch failures before the customers start complaining about the unavailability of the reports/data? Wouldn't it be good to get notified about any long running jobs before these actually become a problem? We will find answers to those problems in this paper.

This paper assumes that you are using Platform Process Manager for scheduling your SAS jobs and your deployment is on UNIX/Linux platforms.

Process Manager updates information about flows and jobs running in it to a log file named history.log<*index*> available under *JS_TOP*/work/history directory. When you want to check the history of a finished flow instance, Flow Manager looks up that history in this directory.

By default, this directory holds 15 days of history.log files and it is governed by the "**JS_HISTORY_CLEAN_PERIOD**" parameter defined in js.conf file. Similarly, when the history log file reaches the maximum size specified in "**JS_HISTORY_SIZE**" (Default=50000 bytes) or the maximum number of hours of data, as specified in "**JS_HISTORY_LIFETIME**" (Default=24 hours) in the js.conf file, a new history log file is created. The numeric suffix of the file increases as each new file is created

CONCEPT

Below is a sample list of log files that will have the information of the jobs running/completed in Flow Manager and are saved under *JS_TOP*/work/history.

```
-rw-r--r-- 1 sas sasusers 500129 Oct  4 05:00 history.log.981
-rw-r--r-- 1 sas sasusers 500028 Oct  4 10:37 history.log.982
-rw-r--r-- 1 sas sasusers 500091 Oct  4 18:30 history.log.983
-rw-r--r-- 1 sas sasusers 500170 Oct  5 05:06 history.log.984
```

A typical information in the history.log file includes name of the flow, owner of the flow, flow id, flow start time/end time, flow status, Job name, Job Id, Job start time/end time, host on which the job was executed and the job status.

Now let us see how to write a SAS program to extract the information from *history.log<index>* files to a SAS dataset.The entire SAS program suite can be divided to four parts

SAS MACRO PROGRAM TO READ THE CONTENT OF HISTORY.LOG<*INDEX*> FILES IN TO A SINGLE INPUT FILE

First we will get the list of history.log files. As our aim is to monitor jobs for the day we will filter out the history.log files that apply for that day. We will then copy the content of each history.log file to a single file which we use it as our input file to create the SAS datasets. Note XCMD has to be enabled in your SAS session for you to use PIPE and CALL SYSTEM routines.

Please read through each comment for better understanding of the code. Below is the SAS macro program:

```
%macro read_hist();

 filename mylist pipe "ls -ltr
/opt/sas/platform/pm/work/history";
  /*please replace /opt/sas/platform/ with the IBM Spectrum LSF
Process Manager Installation directory at your site*/
  /*pipe enables you to invoke a program outside of SAS and
redirect the programs output to SAS */

/*get todays date*/
data _null_;
 call symput ('run_dt',day(today()));
run;
%put &run_dt.;

/*get the file listing of JS_TOP/work/history directory in to a
SAS dataset*/
data myfiles;
 infile mylist lrecl=400 truncover;
 length permis $10 filelink $1 owner $20 group $20 size $8 month
$3 day $2 time $20 filename $200;
 input permis $ filelink $ owner $ group $ size $ month $ day $
time $ filename $;
run;

/*get the filenames for today */
data myfiles (keep=filename);
 set myfiles nobs=obs;
 if day=&run_dt. then output;
run;

data _null_;
 set myfiles end=last;
 call symput('hist_file_nm' || trim(left(put(_n_,8.))),filename);
            /*as you read each filename assign it to variable*/
 if last then call symput('total',trim(left(put(_n_,8.))));
             /*get the total count of history.log files that you
need to read for the day*/
run;
%put &total;
```

```
    /*---copy the content of all the history files to one single
    file---*/

    %let hist_path=/opt/sas/platform/pm/work/history/; /*replace this
    with the PM install directory at your site*/
    %global op_path;
    %let op_path=/tmp/; /*your output location*/

    %do i=1 %to &total;
        %if &i=1 %then
            %do;
               data _null_;
                call system("rm &&op_path.hist_append.txt");
                  /*create a new file, every time the program runs*/
                call system("cp &hist_path.&&hist_file_nm&i
    &op_path.hist_append.txt");
                  /*copy the content to that new file*/
               run;
            %end;
        %else
            %do;
               data _null_;
                call system("cat &hist_path.&&hist_file_nm&i >>
    &op_path.hist_append.txt");
                  /*append the content of further history.log files */
               run;
            %end;
    %end;

%mend read_hist;
%read_hist;
```

After the execution of the program, a file named hist_append.txt will be created under the /tmp/directory.

rw-r----- 1 sas sasusers 1172833 Feb 16 11:19 /tmp/hist_append.txt

DATA STEP TO CREATE A SAS DATASET THAT HAS INFORMATION ABOUT THE FINISHED JOBS

The following is an excerpt from /tmp/hist_append.txt file:

"FLOW" "sas" "1538052600" "117507:sas:DPAI_PM_Monitoring" "**Start flow**" " "

"JOB" "sas" "1538052600" "117507:sas:DPAI_PM_Monitoring:PM_Monitoring" "**Start job**" " "

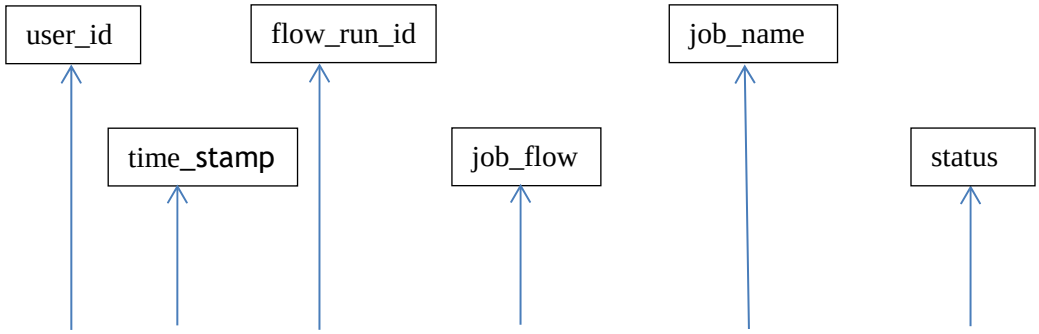
"JOB" "sas" "1538052603" "117507:sas:DPAI_PM_Monitoring:PM_Monitoring" "**Started job**" "JobId=5139"

"JOB" "sas" "1538052608" "117507:sas:DPAI_PM_Monitoring:PM_Monitoring" "**Execute job**" "JobId=5139|Host=gridcompute06.server.domain.com"

"JOB" "sas" "1538052608" "117507:sas:DPAI_PM_Monitoring:PM_Monitoring" "**Finished job**"
JobId=5139|State=Done|Status=0|StartTime=1538052603|FinishTime=153805260
8|CPUUsage=0.192000 sec"

"FLOW" "sas" "1538052608" "117507:sas:DPAI_PM_Monitoring" "**Finished flow**"
"State=Done|Status=0|StartTime=1538052600|FinishTime=1538052608"

In this section we will use /tmp/hist_append.txt as our input file to create a SAS dataset that contains information about the finished jobs.



"JOB" "sas" "1538052608" "117507:sas:DPAI_PM_Monitoring:PM_Monitoring" "**Finished job**"
"JobId=5139|State=Done|Status=0|StartTime=1538052603|FinishTime=1538052608|CPU
Usage=0.192000 sec"

For easy reference, I have mapped the column names against their values. Have used LENGTH=, in combination with the $VARYING. informat, to read a file that contains variable-length records. As SAS executes the first INPUT statement, SAS determines the line length of the record and assign that value to the variable linelen. The single trailing @ holds the record in the input buffer for the next INPUT statement. The second INPUT statement uses the varlen value with the informat $VARYING1500. to read the second variable named rec.

I have extracted the values for other variables using the value of rec column. Also please note that the time values in the history.log files are stored as UNIX epoch time which is the number of seconds from Jan 1, 1970. SAS calculates time value as the number of seconds from Jan 1, 1960. I have therefore converted the UNIX epoch time to SAS time by adding 315619200 to the time. The number **315619200** is the number of seconds between Jan 1, 1970 and Jan 1, 1960.

Below is the DATA STEP:

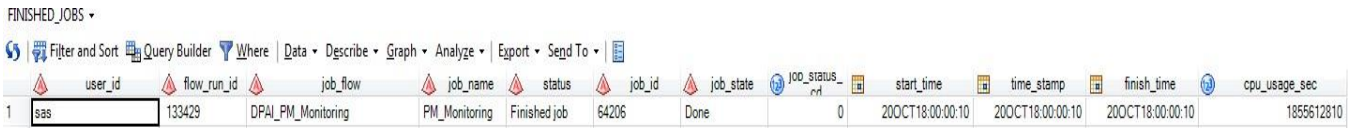
```
filename fnam "&op_path./hist_append.txt";
data finished_jobs(drop= rec firstvar varlen);
  infile fnam length=linelen lrecl=5000;
  length user_id $20. flow_run_id $10. job_flow $40. job_name $100.
status $15. job_id $20. job_state $10. job_status_cd 8. start_time 8.
time_stamp 8. finish_time 8. cpu_usage_sec 8.;
```

```
  format start_time datetime18. time_stamp datetime18. finish_time
datetime18.;

  input firstvar $ 1-1 @;
  varlen=linelen-1;
  input @2 rec $varying1500. varlen;
  /* convert UNIX epoch time to SAS time */
  time_stamp=input(scan(rec,5,'"'),20.)+315619200;
  status=translate(scan(rec,9,'"'),'','"');
  if status='Finished job' and (datepart(time_stamp)=today()) then
   do;
     user_id=translate(scan(rec,2," "),'','"');
     flow_run_id=translate(scan(scan(rec,4," "),1,':'),'','"');
     job_flow=translate(scan(scan(rec,4," "),3,':'),'','"');
     job_name=translate(scan(scan(rec,4," "),4,':'),'','"');
     job_state=scan(scan(scan(rec,11,'"'),2,'|'),2,'=');
     job_id=scan(scan(scan(rec,11,'"'),1,'|'),2,'=');
     job_status_cd=input(scan(scan(scan(rec,11,'"'),3,'|'),2,'='),4.);
     start_time=input(scan(scan(scan(rec,11,'"'),4,'|'),2,'='),20.)
+315619200;    /* convert UNIX epoch time to SAS time*/
     finish_time=input(scan(scan(scan(rec,11,'"'),5,'|'),2,'='),20.)
+315619200;
     cpu_usage_sec=input(scan(scan(scan(rec,11,'"'),5,'|'),2,'='),20.)
+315619200;
     output;
   end;
run;
```

The output SAS dataset then would look like that in Figure 1.

Figure 1: FINISHED_JOBS Dataset



DATA STEP TO CREATE A SAS DATASET THAT HAS INFORMATION ABOUT THE STARTED JOBS

In this section we will use /tmp/hist_append.txt as our input file to create a SAS dataset that contains information about the jobs that have started execution.

"JOB" "sas" "1538052608" "117507: sas:DPAI_PM_Monitoring:PM_Monitoring" "**Execute job**" "JobId=5139|Host=gridcompute06.server.domain.com"

We can create a dataset following a similar logic to what we used in the above DATA step. Below is the DATA STEP:

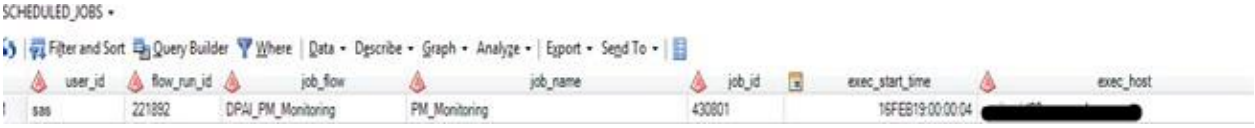
```
filename fnam "&op_path./hist_append.txt";
data scheduled_jobs(drop= rec firstvar varlen status);
  infile fnam length=linelen lrecl=5000;
```

```
  length user_id $20. flow_run_id $10. job_flow $40. job_name $100.
job_id $20. exec_start_time 8. exec_host $30.;
  format exec_start_time datetime18.;

  input firstvar $ 1-1 @;
  varlen=linelen-1;
  input @2 rec $varying1500. varlen;
  exec_start_time=input(scan(rec,5,'"'),20.)+315619200;
  status=translate(scan(rec,9,'"'),'','"');
  if status='Execute job' and (datepart(exec_start_time)=today()) then
   do;
     user_id=translate(scan(rec,2," "),'','"');
     flow_run_id=translate(scan(scan(rec,4," "),1,':'),'','"');
     job_flow=translate(scan(scan(rec,4," "),3,':'),'','"');
     job_name=translate(scan(scan(rec,4," "),4,':'),'','"');
     job_id=scan(scan(scan(rec,11,'"'),1,'|'),2,'=');
     exec_host=scan(scan(scan(rec,11,'"'),2,'|'),2,'=');
     output;
   end;
run;
```

The output SAS dataset then would look like that in Figure 2.

Figure 2: SCHEDULED_JOBS Dataset



COMBINE THE DATASETS CREATED TO CREATE A FINAL DATASET WHICH WILL HAVE ALL THE INFORMATION ABOUT THE JOBS

In this section, we will use PROC SQL to join the datasets finished_jobs and scheduled_jobs.

The output dataset job_details will have the start_time, finish_time, run_time, execution host, status and many more information about the jobs.

```
proc sql noprint;
  create table job_details
   as
    select
       datepart(exec_start_time) as run_date format=date8.,
       a.*,
       b.finish_time,
       b.job_state,
       b.job_status_cd,
       (b.finish_time - a.exec_start_time)/60 as run_time
    from scheduled_jobs a
       left outer join
         finished_jobs b
       on a.flow_run_id=b.flow_run_id and a.job_id=b.job_id
```

```
    order by exec_start_time;
run;
```

The resulting output SAS dataset would look like that in Figure 3.

Figure 3: JOB_DETAILS Dataset

MONITORING THE JOBS

Our aim is to identify failed and long running jobs. We will schedule the above SAS program to run every 30 minutes and will add the below codes to identify the failed and long running jobs.

As you know, jobs that have run successfully will have an exit code of 0 and a job which has ran successfully but with some warnings will have an exit code of 1. So, in order to capture only the failed jobs we will search for all finished jobs that have an exit code of 2 and above.

Below PROC SQL statement will gather information about all the failed jobs in the last 30 minutes:

```
proc sql  noprint;
create table failed_jobs
 as
   select
          user_id,
          flow_run_id,
          job_flow,
          job_name,
          status,
          job_id,
          job_state,
          job_status_cd,
          start_time,
          finish_time,
          cpu_usage_sec
    from finished_jobs
    where job_status_cd >= 2 and timepart(finish_time)> time()- 1800;
run;
```

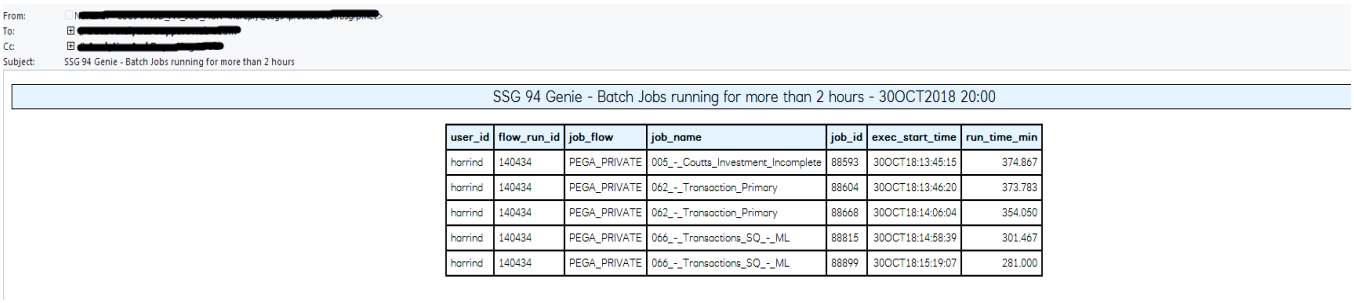
Similarly, we will use the job_details dataset to identify long running jobs. Below PROC SQL statement will identify the jobs that are running for more than 2 hours:

```
proc sql noprint;
create table long_run_jobs
  as
    select
```

```
            user_id,
            flow_run_id,
            job_flow,
            job_name,
            job_id,
            exec_start_time,
            (time()-timepart(exec_start_time))/60 as run_time_min
      from job_details
      where finish_time is null and timepart(exec_start_time)< time()-
7200;
run;
```

You can then include a SAS code to email you or your team with the details of failed and long running jobs. In this paper, I have not included the code for sending email; however I have given the sample emails that we get from our job.

Result 1: Sample email output that lists long running jobs

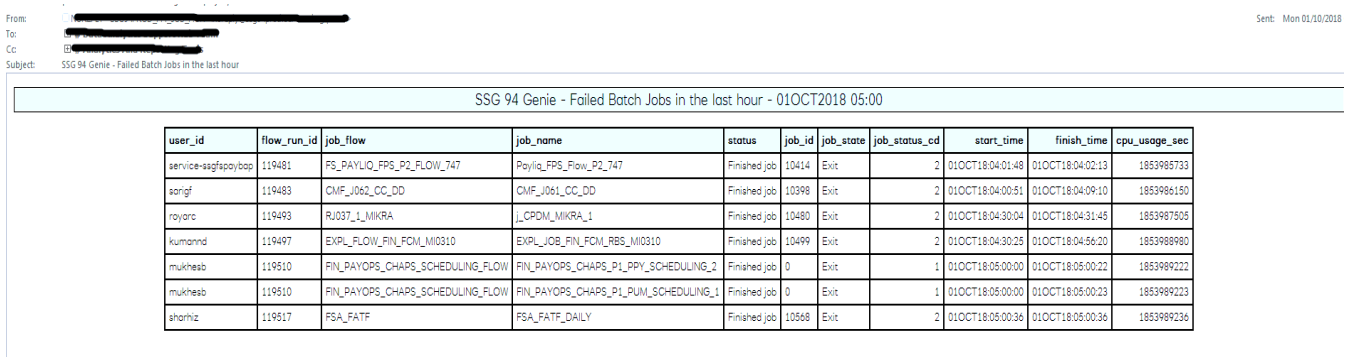


From: [redacted]
To: [redacted]
Cc: [redacted]
Subject: SSG 94 Genie - Batch Jobs running for more than 2 hours

SSG 94 Genie - Batch Jobs running for more than 2 hours - 30OCT2018 20:00

user_id	flow_run_id	job_flow	job_name	job_id	exec_start_time	run_time_min
harrind	140434	PEGA_PRIVATE	005_-_Coutts_Investment_Incomplete	88593	30OCT18:13:45:15	374.867
harrind	140434	PEGA_PRIVATE	062_-_Transaction_Primary	88604	30OCT18:13:46:20	373.783
harrind	140434	PEGA_PRIVATE	062_-_Transaction_Primary	88668	30OCT18:14:06:04	354.050
harrind	140434	PEGA_PRIVATE	066_-_Transactions_SQ_-_ML	88815	30OCT18:14:58:39	301.467
harrind	140434	PEGA_PRIVATE	066_-_Transactions_SQ_-_ML	88899	30OCT18:15:19:07	281.000

Result 2: Sample email output that lists failed jobs



From: [redacted]
To: [redacted]
Cc: [redacted]
Subject: SSG 94 Genie - Failed Batch Jobs in the last hour
Sent: Mon 01/10/2018

SSG 94 Genie - Failed Batch Jobs in the last hour - 01OCT2018 05:00

user_id	flow_run_id	job_flow	job_name	status	job_id	job_state	job_status_cd	start_time	finish_time	cpu_usage_sec
service-ssgfspaybap	119481	FS_PAYLIQ_FPS_P2_FLOW_747	Payliq_FPS_Flow_P2_747	Finished job	10414	Exit	2	01OCT18:04:01:48	01OCT18:04:02:13	1853985733
sorigf	119483	CMF_J062_CC_DD	CMF_J061_CC_DD	Finished job	10398	Exit	2	01OCT18:04:00:51	01OCT18:04:09:10	1853986150
royarc	119493	RJ037_1_MIKRA	J_CPDM_MIKRA_1	Finished job	10480	Exit	2	01OCT18:04:30:04	01OCT18:04:31:45	1853987505
kumarnd	119497	EXPL_FLOW_FIN_FCM_MI0310	EXPL_JOB_FIN_FCM_RBS_MI0310	Finished job	10499	Exit	2	01OCT18:04:30:25	01OCT18:04:56:20	1853988980
mukhesb	119510	FIN_PAYOPS_CHAPS_SCHEDULING_FLOW	FIN_PAYOPS_CHAPS_P1_PPY_SCHEDULING_2	Finished job	0	Exit	1	01OCT18:05:00:00	01OCT18:05:00:22	1853989222
mukhesb	119510	FIN_PAYOPS_CHAPS_SCHEDULING_FLOW	FIN_PAYOPS_CHAPS_P1_PUM_SCHEDULING_1	Finished job	0	Exit	1	01OCT18:05:00:00	01OCT18:05:00:23	1853989223
sharhiz	119517	FSA_FATF	FSA_FATF_DAILY	Finished job	10568	Exit	2	01OCT18:05:00:36	01OCT18:05:00:36	1853989236

CONCLUSION

By having the above code running in your scheduler, you will be able to save lot of time and will get notified well in advance about the failed and long running jobs. You can also create a DataMart which contains information about all of the jobs running in Process Manager for all days; which you can use later for any batch job performance analysis you may require.

RECOMMENDED READING

SAS® 9.4 Language Reference: Concepts, Sixth Edition

SAS® 9.4 DATA Step Statements: Reference

SAS® Note available at http://support.sas.com/kb/31/756.html

More information about history.log file is available at https://www.ibm.com/support/knowledgecenter/en/SSZSHQ_10.1.0/source/history.log.n.5.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Pradeep Ponnusamy
Royal Bank of Scotland Group
Gogarburn, Edinburgh EH12 9BH
United Kingdom
+44 7741036734
pradeep.ponnusamy@rbs.co.uk