SESUG Paper 27-2021

This Week's Forecast: Read From The Cloud How To Collect Records Read From RDBMS

ABSTRACT

Many data centers are considering and planning for moves to cloud storage at some point in the future. Cloud providers charge by the number of records read from the cloud which is much different than previous sizing exercises. This paper will detail how I approached the task of collecting records read from Oracle and Netezza from the SAS environment.

INTRODUCTION

Planning for cloud storage integration or cloud processing with SAS is a multi-faceted process. There are different configurations including hybrids of cloud and on premises to consider. I found that I could not even begin to address what we need in the future if I don't have granular statistics of what goes on in the current SAS environment.

Cloud providers charge a price by records read from the cloud. I needed to find out what kind of price tag that would look like in my current environment which brought me to the question "How do I track when users connect from the SAS environment into RDBMS and read records?" The administrators for the RDBMS at my site were unable to take on this task for me. Therefore, I was pleased to realize that I had all I needed to collect this information with SAS options; specifically, SASTRACE.

The SAS environment discussed in this paper is 3 tier (Metadata, Compute, MidTier) set up on Linux. An estimated 95% of jobs were captured using this process. Batch jobs located in user locations amid the server were omitted.

Keep in mind that this paper is meant to be a general guide; other environments will need to look for different key words or have options set where it makes the most sense for that environment.

EVALUATING THE CURRENT SAS ENVIRONMENT

There are a few planning questions to evaluate before taking the next steps.

1. How do users connect to the SAS environment?
2. Are all user logs captured on the SAS server?
3. Are production run SAS logs captured?
4. How are connections made to RDBMS?
5. What type of data is available to aggregate counts into different categories?
6. How do I need to present the results to technology partners within the organization?

The following are answers to the questions based on my site's SAS environment.

HOW DO USERS CONNECT TO THE SAS ENVIRONMENT?

In my environment there are three main ways users connect to SAS:

1. SAS Enterprise Guide: WorkspaceServer
2. PC SAS: Foundation or ConnectServer
3. Scheduled SAS: BatchServer

ARE ALL USER LOGS CAPTURED ON THE SAS SERVER?

Our SAS Enterprise Guide connects directly to the server with all logs for the SAS environment going to (filesystem)/config/Lev1/Logs.

PC SAS users were still connecting with Foundation SAS via a tcpunix script. This connection uses (filesystem)/binary/SASFoundation/9.4/sas. I worked with SAS Technical Support to get user sessions connected using the ConnectServer so that logs would be retained on the server at (filesystem)/config/Lev1/Logs. (this is the same location as SAS Enterprise Guide logs.)

User batch jobs are not directed to one single repository on the SAS compute tier.

ARE PRODUCTION RUN SAS LOGS CAPTURED?

Our production batch SAS jobs connect through (filesystem)/config/Lev1/SASApp/BatchServer and are invoked by a third party scheduler that runs an agent on the SAS compute tier. Those logs are directed to a separate location on the server that is designated for SAS production batch jobs.

HOW ARE CONNECTIONS MADE TO RDBMS?

My users utilize three methods to connect into RDBMS:

1. Pre-assigned libnames via SAS Management Console (Oracle only)
2. Hardcoded libname statements (Oracle and Netezza)
3. Pass Thru design (Oracle and Netezza)

WHAT TYPE OF DATA IS AVAILABLE TO AGGREGATE COUNTS INTO DIFFERENT CATEGORIES?

Any information that is entered into SAS Management Console when a new user is added into the SAS system can be pulled from SAS metadata with a macro provided by SAS.

Display Name, User ID, and Title will be the data elements that I will pull out of SAS metadata to use in the aggregate reporting summary.

HOW DO I NEED TO PRESENT THE RESULTS TO TECHNOLOGY PARTNERS WITHIN THE ORGANIZATION?

Based on the information available to me, I am able to report by the following:

- Total reads by system (Oracle or Netezza)
- Total reads by user (Name, Display Name)
- Total reads by department (Title)
- Total reads by SAS job (Log name)

USING SASTRACE IN THE SAS ENVIRONMENT

Now that we've determined which logs need to be collected, the next step is getting the necessary information into the logs. SASTRACE is the system option that brings detail into the log (trace information) from a database engine.

Based on much trial and error, the best fit to extract the information I need from both Oracle and Netezza was the following options statement:

```
options SASTRACE=',,t,dsab' SASTRACELOC=SASLOG NOSTSUFFIX;
```

Here is a snippet from SAS documentation (link found in the Recommended Reading section of this paper) which details the options that I set for the environment:

Syntax

SASTRACE=',,,d' | ',,d,' | 'd,' | 'd,,,' | ',,,db' | ',,,s' | ',,,sa' | ',,t,' | OFF

Required Arguments

',,,d'
specifies that all SQL statements that are sent to the DBMS are sent to the log. Here are the applicable statements:

SELECT	DELETE
CREATE	SYSTEM CATALOG
DROP	COMMIT
INSERT	ROLLBACK
UPDATE	

For engines that do not generate SQL statements, API calls and all parameters are sent to the log.

',,d,'
specifies that all routine calls are sent to the log. All function enters, exits, and pertinent parameters and return codes are traced when you select this option. The information varies from engine to engine, however.

This option is most useful if you have a problem and need to send a SAS log to technical support for troubleshooting.

'd,'
specifies that all DBMS calls (such as API and client calls, connection information, column bindings, column error information, and row processing) are sent to the log. This information varies from engine to engine, however.

This option is most useful if you have a problem and need to send a SAS log to technical support for troubleshooting.

'd,,,'
specifies that version information for the current DBMS and client are displayed in the log.

',,,db'
specifies that only a brief version of all SQL statements that the ',,,d' option normally generates are sent to the log.

',,,s'
specifies that a summary of timing information for calls made to the DBMS is sent to the log.

',,,sa'
specifies that timing information for each call that is made to the DBMS is sent to the log along with a summary.

',,t,'
specifies that all threading information is sent to the log. Here is the information that it includes:

- number of threads that are spawned
- number of observations that each thread contains
- exit code of the thread, if it fails

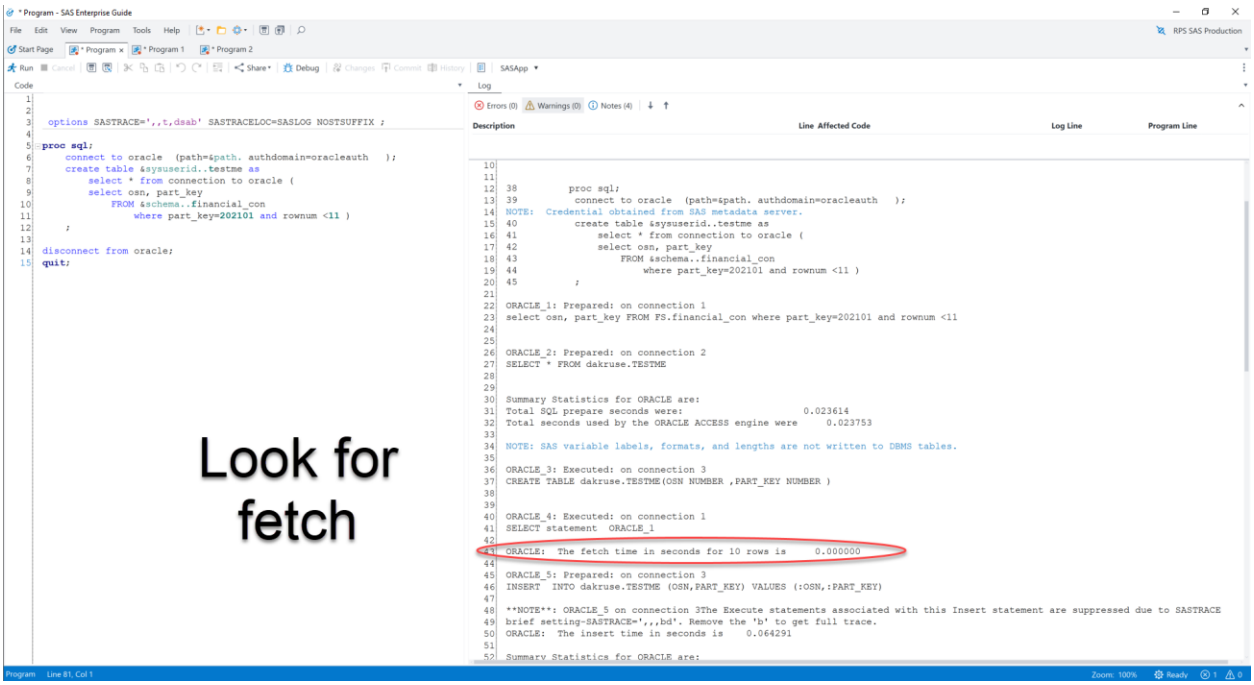
I placed the option in (filesystem)/config/Lev1/SASApp/appserver_autoexec_usermods.sas. This can easily be commented out to back out the option. There is no need to restart any services in the SAS environment for this option to be turned on or off.

PASS THRU CODE

Knowing how your users code will determine how you capture the data you need in the log. My users often code Oracle pass thru with the following syntax, so using these SAS trace settings, I was able to capture the FETCH messages which contain record count. If using an EXECUTE statement, the FETCH statement would not appear in the log and alternate SASTRACE options may be necessary.

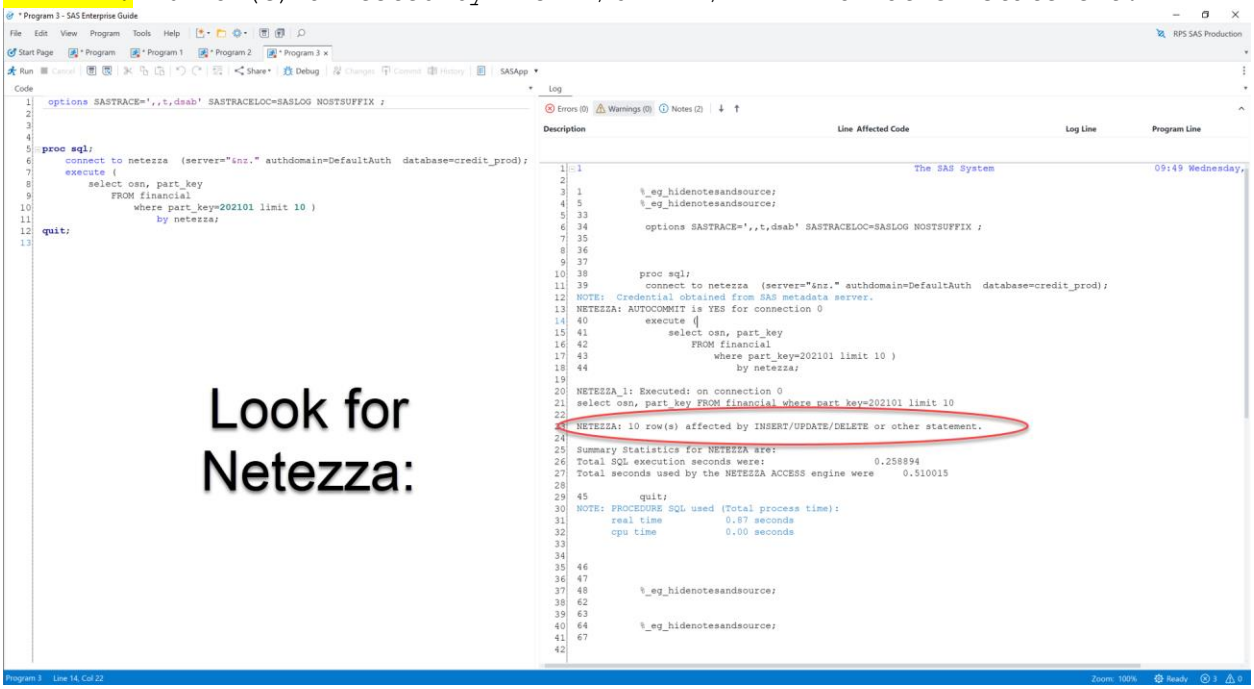
Here is an example of the detail the SASTRACE settings brought into our logs for Oracle code:

```
ORACLE:  The fetch time in seconds for 10 rows is      0.000000
```



Here is an example of the detail the SASTRACE settings brought into our logs for Netezza code:

NETEZZA: 10 row(s) affected by INSERT/UPDATE/DELETE or other statement.



CONTROLLING THE SIZE OF THE LOGS

Turning on SASTRACE is a processing and storage load addition to the environment. Make sure there is enough physical space for the log storage on the filesystem and consider adding additional programming options to reduce the messages written into the log.

The default is for SAS to dynamically assign the number of records for the read and write buffer. When reading millions of records and seeing 180 record chunks be grabbed at a time, this makes for an extremely large log. Therefore, setting the buffers can reduce the number of FETCH or INSERT notes written to the log thereby reducing log size. There is a tradeoff with available memory so you may have to make some adjustments for certain users or processes on the fly. I only had one user that ran into memory issues with the settings I chose for 32767. I recommended the user set (for that particular program) the buffer to 10000 instead and the memory problem resolved.

SASTRACE can also be turned off at the program level, but I did not advertise this to users as I assumed they would turn it off instead of trying to reduce the number of records read to the buffer for problem resolution. To turn this option off at the programming level, add this code into the program:

```
options sastrace=off ;
```

The following Oracle libname assignments were modified to add READBUFF and/or INSERTBUFF depending on the location of the schema.

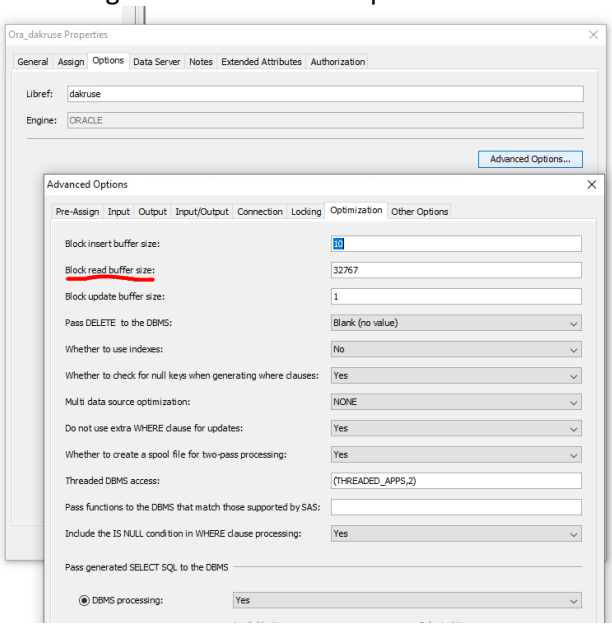
Hardcoded Examples:

```
LIBNAME FS ORACLE PATH=&path. AUTHDOMAIN=OracleAuth SCHEMA=FS READBUFF=32767;
/*ORACLE LIBNAME READ ONLY */

LIBNAME &sysuserid. ORACLE PATH=&path. AUTHDOMAIN=OracleAuth
SCHEMA=&sysuserid. INSERTBUFF=32767  READBUFF=32767; /*ORACLE LIBNAME READ /
WRITE  */
```

If your environment contains pre-assigned libnames from SAS Management Console, use the properties pallet of the libname assignment to adjust the buffer record count. (PROPERTIES –> OPTIONS –> ADVANCED OPTIONS –> OPTIMIZATION)

SAS Management Console Example:



The results of this change are the maximum value of 32767 (according to the option set) was grabbed at time of execution.

ORACLE: The fetch time in seconds for 32767 rows is 0.000000

The following Netezza libname assignments were modified to add ROWSET_SIZE and/or INSERTBUFF depending on the location of the schema.

```
LIBNAME cr_prd netezza server=&NZ_SERVER. AUTHDOMAIN=DefaultAuth
Database=credit_prod DIRECT_SQL=YES ROWSET_SIZE=32767; /*NETEZZA LIBNAME READ
ONLY */
LIBNAME SANDBOX netezza server=&NZ_SERVER. AUTHDOMAIN=DefaultAuth
Database=SANDBOX DIRECT_SQL=YES ROWSET_SIZE=32767 INSERTBUFF=32767; /*NETEZZA
LIBNAME READ / WRITE  */
```

Notice the results of this change are NOT the maximum value of 32767 (according to the option set), but the maximum number the query pulled was grabbed at time of execution. The option didn't leave the amount of records for SAS to decide, it forced the maximum number for that particular query. This scenario also reduces the number of notes to the SAS log.

NETEZZA: Fetch time in seconds for 3657 rows is 0.008666

PROGRAMMING WITH BASE SAS TO PARSE THE LOGS

Once the detailed information is in the logs and on the server, it is time to create the program to parse the information out of the log and create a report. I will provide some snippets of SAS code in this paper to get you started. The full code from the program I use will not be provided because it cannot be directly applied to any environment. Use the SAS snippets below to construct a program that works for your environment.

GATHERING LOG LISTING FOR A WINDOW OF TIME

I used a DATA _NULL_ step to execute UNIX system commands for a seven-day period. I scraped the log for the exact syntax that will appear in the log (for accessing RDBMS) and directed the log names to one file that will be read in by SAS in subsequent steps.

The results of this system command provide the log name followed by the syntax that was matched inside the log.

Display 1. Code Used To Scrape Logs

```
/*Scan for logs that accessed RDBMS*/
%LET start_dt = 2021-06-20;
%LET end_dt   = 2021-06-27;

data _null_;
call system("cd /opt/sas/config/Lev1/Logs"); *Server log location of most sessions ;
call system("grep -n 'ORACLE:  The fetch \| row(s) affected \|NETEZZA: Fetch
time ' $(find . -type f -newermt &start_dt.  ! -newermt &end_dt.  -name
'*.log') > /opt/sas/users/dakruse/tracking_logs.txt");
run;
```

Display 2. Example Output From Resulting Text File

The log file name is in the orange rectangle and the matched syntax from the system command is highlighted in red.

```
./SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:4531:NETEZZA: Fetch
time in seconds for 32767 rows is    0.034733
./SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:4532:NETEZZA: Fetch
time in seconds for 32767 rows is    0.028621
./SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:4533:NETEZZA: Fetch
time in seconds for 32767 rows is    0.029745
./SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:4534:NETEZZA: Fetch
time in seconds for 7058 rows is    0.103147
./SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:4535:NETEZZA: Fetch
time in seconds for 0 rows is    0.000019
./SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4400:ORACLE:  The
fetch time in seconds for 250 rows is      0.000000
./SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4401:ORACLE:  The
fetch time in seconds for 250 rows is      0.000000
./SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4402:ORACLE:  The
fetch time in seconds for 250 rows is      0.000000
./SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4403:ORACLE:  The
fetch time in seconds for 250 rows is      0.000000
```

READ TEXT FILE INTO SAS DATA STEP TO PARSE FIELDS

Using the text file as input, the goal is to parse the items that we want and put them into variables. In the code below, I pull out the log name (LOG_NM), the Oracle record count (ora_cnt) and the Netezza record count (nz_cnt) from each record in the text file. The other fields can be discarded in later steps as they are only used to calculate the location of the necessary information (counts).

Display 3. Data Step Code Used To Parse File

```
37  filename rec_read '/opt/sas/SAS_env/users/dakruse/tracking_logs.txt';
38
39  data summary_read;
40  format line_itm  $250. ora_pos nz_pos  nz_pos2 8. ora_cnt nz_cnt $20. LOG_NM $60. ;
41   infile rec_read truncover  ;
42   input  @'./' line_itm $250.;
43
44   LOG_NM=scan(line_itm,1,':') ;
45
46   ora_pos=indexw(line_itm,'fetch time in seconds for ');
47
48    if index(line_itm,'NETEZZA:') = 0
49      then nz_pos=index(line_itm,'NETEZZA') ;
50      else nz_pos2=index(line_itm,'NETEZZA:');
51
52   if ora_pos >0 then
53      ora_cnt=substr(line_itm,ora_pos+26);
54      else ora_cnt='n/a' ;
55   if nz_pos >0 then
56      nz_cnt=substr(line_itm,nz_pos+9);
57      else nz_cnt='n/a' ;
58    if nz_pos2 >0 then
59      nz_cnt=scan(substr(line_itm,nz_pos2+35),1);
60      else nz_cnt='n/a' ;
61  run;
```

Display 4. Results From Data Step Parse

Example of records read from Netezza:

- line_itm value (illustrated with the red rectangle) contains the entire row of data from the text file.
- nz_cnt value (illustrated with the blue rectangle) contains the number of records parsed from that line of the log.
- LOG_NM value (illustrated with light orange rectangle) contains the name of the SAS log

	line_itm	ora_pos	nz_pos	nz_pos	ora_cnt	nz_cnt	LOG_NM
1	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log:274:NETEZZA: Fetch time in seconds for 11388 rows is 0.011987	0	.	55	n/a	11388	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log
2	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log:281:NETEZZA: Fetch time in seconds for 0 rows is 0.000020	0	.	55	n/a	0	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log
3	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log:328:NETEZZA: Fetch time in seconds for 11388 rows is 0.012641	0	.	55	n/a	11388	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log
4	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log:329:NETEZZA: Fetch time in seconds for 0 rows is 0.000018	0	.	55	n/a	0	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log
5	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log:428:NETEZZA: Fetch time in seconds for 18 rows is 0.000032	0	.	55	n/a	18	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log
6	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log:429:NETEZZA: Fetch time in seconds for 0 rows is 0.000019	0	.	55	n/a	0	SASApp_ConnectServer_2021-06-20_rpsas2p_11917.log
7	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:209:NETEZZA: Fetch time in seconds for 32767 rows is 0.001088	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
8	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:606:NETEZZA: Fetch time in seconds for 32767 rows is 0.112353	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
9	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:625:NETEZZA: Fetch time in seconds for 32767 rows is 0.145572	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
10	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:630:NETEZZA: Fetch time in seconds for 32767 rows is 0.142677	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
11	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:635:NETEZZA: Fetch time in seconds for 32767 rows is 0.148244	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
12	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:640:NETEZZA: Fetch time in seconds for 32767 rows is 0.152985	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
13	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:645:NETEZZA: Fetch time in seconds for 32767 rows is 0.153942	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
14	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:650:NETEZZA: Fetch time in seconds for 32767 rows is 0.142111	0	.	55	n/a	32767	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log
15	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log:655:NETEZZA: Fetch time in seconds for 336 rows is 0.001497	0	.	55	n/a	336	SASApp_ConnectServer_2021-06-20_rpsas2p_26769.log

Example of records read from Oracle:

- line_itm value (illustrated with the red rectangle) contains the entire row of data from the text file.
- ora_cnt value (illustrated with the blue rectangle) contains the number of records parsed from that line of the log followed by some extra information which should be parsed out further.
- LOG_NM value (illustrated with light orange rectangle) contains the name of the SAS log

	line_itm	ora_pos	nz_pos	nz_pos	ora_cnt	nz_cnt	LOG_NM
1	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4400:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
2	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4401:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
3	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4402:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
4	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4403:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
5	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4404:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
6	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4405:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
7	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4406:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
8	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4407:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
9	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4408:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
10	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4409:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
11	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4410:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
12	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4411:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
13	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4412:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
14	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4413:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log
15	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log:4414:ORACLE: The fetch time in seconds for 250 rows is 0.000000	70	0	.	250 rows is 0.00	n/a	SASApp_WorkspaceServer_2021-06-20_rpsas2p_3995.log

SCRAPING LOG FILES TO RETRIEVE USER INFORMATION

A second pass at the same logs is now necessary to retrieve the user ID (for reporting purposes). I rerun the system command I ran before but I change the syntax that I need to capture. In this case, our logs have the keyword ESMUSER followed by the user ID which ran the process. Similar to the first scrape, I will output these results into another text file read into SAS in a subsequent step of the program.

Display 5. Code To Capture User ID

```
data _null_;
call system("cd /opt/sas/config/Lev1/Logs");
call system("grep -n 'ESMUSER is ' $(find . -type f -newermt &start_dt. ! -
newermt &end_dt. -name '*.log') >
/opt/sas/users/dakruse/tracking_logs_user.txt");
run;
```

Display 6. Results From Data Scrape Of User ID

The second scrape of the log brings in the key word "ESMUSER" followed by the user ID.

```
./SASApp_WorkspaceServer_2021-06-20_rpsas2p_17907.log:81:ESM: ESMUSER is dakruse
./SASApp_WorkspaceServer_2021-06-20_rpsas2p_18034.log:81:ESM: ESMUSER is usr1234
./SASApp_WorkspaceServer_2021-06-20_rpsas2p_18279.log:81:ESM: ESMUSER is usr9876
./SASApp_WorkspaceServer_2021-05-31_rpsas2p_3515.log:79:ESM: ESMUSER is usrnew1
```

Display 7. Code To Capture User ID

```
filename loglist '/opt/sas/users/dakruse/tracking_logs_user.txt';

data log_list;
format LOG_NM $60. USER $10.;
 infile loglist truncover  ;
 input  @'./' LOG_NM $60.
            @'is ' USER $10. ;
LOG_NM=scan(LOG_NM,1,':') ;
 run;
```

Display 8. Results From Scraping User ID

	LOG_NM	USER
1	SASApp_WorkspaceServer_2021-06-20	dakruse
2	SASApp_WorkspaceServer_2021-06-20	usr1234
3	SASApp_WorkspaceServer_2021-06-20	usr9876
4	SASApp_WorkspaceServer_2021-05-31	usrnew1

You can add user ID to the logs if there is no current keyword to use for parsing. Use a %put statement with your own keyword and the system macro variable of &sysuserID. This can be placed in the same module SASTRACE is controlled: (filesystem)/config/Lev1/SASApp/appserver_autoexec_usermods.sas

Display 9. Adding User ID To The Log

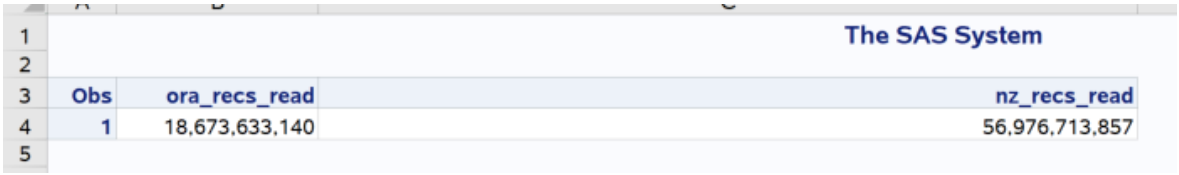
```
1  %put USERID:  &SYSUSERID. ;
27   %put USERID:  &SYSUSERID. ;
USERID:  dakruse
```

BRINGING THE INFORMATION TOGETHER FOR REPORTING

Now is the time to think about how the data needs to be delivered in your organization. If details of a username (not just ID) are important, that can be added with a macro program developed by SAS (provided Display Name is populated in SAS Metadata). Otherwise, the data sets created from the text files can be joined by the LOG_NM field giving you a main data set to work with for aggregating data.

There is a bit more data cleansing to do on the Oracle count (ora_cnt) field from Display 4. Once that number is parsed out of the field, a PROC SQL query with a sum function can provide the total number of records read for the window of time designated by the log capture. We used a seven-day window of time in this example.

Display 10. Total Record Counts For Each RDBMS For 7 Days



The SAS System

Obs	ora_recs_read	nz_recs_read
1	18,673,633,140	56,976,713,857

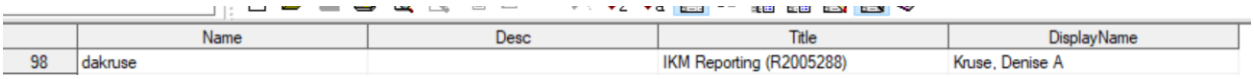
The macro developed by SAS will take information populated in SAS metadata and put it into SAS datasets. Run it as the unrestricted user for a successful execution. The macro resides in *SAS-installation-directory*/SASFoundation/9.4/sasautos.

Display 11. Executing %MDUEXTR Macro

```
options metaserver="MyLinuxServer" metaport=8561 metauser="sasadm@saspw"
metapass="Password" metaprotocol=bridge metarepository=Foundation;
%mduextr(libref=work);
```

The dataset I use from the macro results is called PERSON_INFO. You will see from the results below that data can be joined by user ID to pull back Display Name and Title if desired.

Display 12. Results from %MDUEXTR macro PERSON_INFO data set



	Name	Desc	Title	DisplayName
98	dakruse		IKM Reporting (R2005288)	Kruse, Denise A

Display 13. Snippet of Final Report

The SAS System

Obs	ora_recs_read	nz_recs_read
1	18,673,633,140	56,976,713,857

The SAS System

Obs	user1	Title	DisplayName	ora_recs_read	nz_recs_read
1				.	1,507,932
2		CC RPS		4,227,263,674	1,871,359,191
3		CC RPS		726	.
4		RPS_FIN_		8,100,049	632
5		RPS-Card Underw		22,031,548	38,897,006
6		CC RPS USB		56,562,082	.
7		RPS-Operational Risk		416,219,838	.
8		RPS Shared Services_I		4,019	93,218
9		CC RPS Asst Product Mgr I		3,623	.
10		CC RPS OC Data Analytics		929,306,867	1,794,019,344

CONCLUSION

Records read from RDBMS from the SAS environment was a significant planning point for evaluating a move into the cloud for data storage. I have reviewed some of the factors to consider about your SAS environment when trying to assess the environment that you manage.

We looked at the way connections are made into the SAS environment as well as how to capture those connections through logs with SASTRACE. Keep in mind there are many variations of connecting into a

SAS environment as well as many variations for SASTRACE usage. You will need to analyze and test for your needs.

After all the data is collected into text files, it is a matter of bringing that data into BASE SAS to parse out and aggregate what is needed for reporting purposes.

ACKNOWLEDGMENTS

I'd like to thank David Steves for getting me started with the UNIX find command and syntax which allowed me to collect the logs in the timeframe to be analyzed.

Many thanks to my colleagues Patrick Ryan, David Steves and Donalee Wanna who took the time to review my paper and provide feedback.

RECOMMENDED READING

- *SASTRACE documentation at* https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/acreldb/n0732u1mr57ycrn1urf24gzo38sc.htm
- *%MDUEXTR documentation at*

  https://documentation.sas.com/doc/en/bicdc/9.4/bisecag/n024i4nqa5b12qn1lfek77h69ns5.htm

  https://blogs.sas.com/content/sgf/2016/01/13/sas-administrators-tip-keeping-track-of-sas-users/

CONTACT INFORMATION

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

Denise A. Kruse
U.S. Bank
denise.kruse@usbank.com