PharmaSUG China 2016-31

A Grid Computing Tool to Batch Run a List of SAS® Programs

Huashan Huo, Zhongyu Li, Lu Zhang, PPD, Beijing

ABSTRACT

In clinical research, it is very common that a large number of SAS programs are to be repeatedly batch run due to program modifications or new data updates. In the past few years, several papers authored by pharmaceutical industry programmers (Gilbert Chen 2002; Shu 2006; Prescod Cawley 2010; Wong Sun 2010; Conover 2011; Andrew E. Hansen 2013; Huashan Huo, Fanyu Li 2015) were published in this area to describe methods and tools for automating this process. The purpose of this paper is to introduce a tool for grid computing a large number of SAS programs concurrently. The SAS computing tasks are distributed among multiple computers on a network, all under the tool's self-management. Not only workloads are distributed across a grid of computers, but also overall execution time is geometrically reduced. This paper also describes how the tool supports Multi-User Workload Balancing and Parallel Workload Balancing without SAS/Share and SAS Grid Manager.

INTRODUCTION

Generally speaking, batch run is usually tedious and time consuming, due to the great amount of data and large number of SAS programs. In addition, the widely accepted double programming, may approximately double the workload. Batch run is more efficient than running each of the tasks manually, however, given the fact that batch run is only execution in sequential on a single machine, it still takes us too long to perform all of the tasks.

This paper introduces a tool that makes distributed computational resources available to accomplish a target project. Parallel computations connected by internet can be taken place independently in different hosts. Resources shared in the grid can dramatically reduce the processing time of the whole project. Thus, the total execution time is negatively correlated with number of computers involved in. In an extreme case, it can be as short as the longest execution time of a single SAS program.

In our previous paper we introduced a similar tool built on user-friendly interface (GUI) system, which is capable of running multiple SAS session concurrently in a stand-alone computer and display real-time progress and status information. As soon as fulfilling all the tasks, an automatic wrap-up email will be sent to the leader and corresponding program authors. Customized list filter is available as well for program re-running. By maximizing the utilization of resource of a single computer, the computation time of our previous tool can be increased by two times or more compared to ordinary batch run. However, it is still limited by the computer's own CPU resource and memory capacity. Our existing tool carries on all the advantages, and further improves the computation speed geometrically by coupling distributed virtual computers. Instead of static pre-specified SAS sessions, this tool can launch new SAS session dynamically depending on each computer's own resource.

METHODS

Job scheduling algorithm

Precedency constraints for set of parallel tasks must be guaranteed by executing predecessor tasks before the successor ones. To achieve this goal, we will group the tasks and assign each of them with priority level. The task list along with priority information recorded in excel file will be imported when the tool is launched.

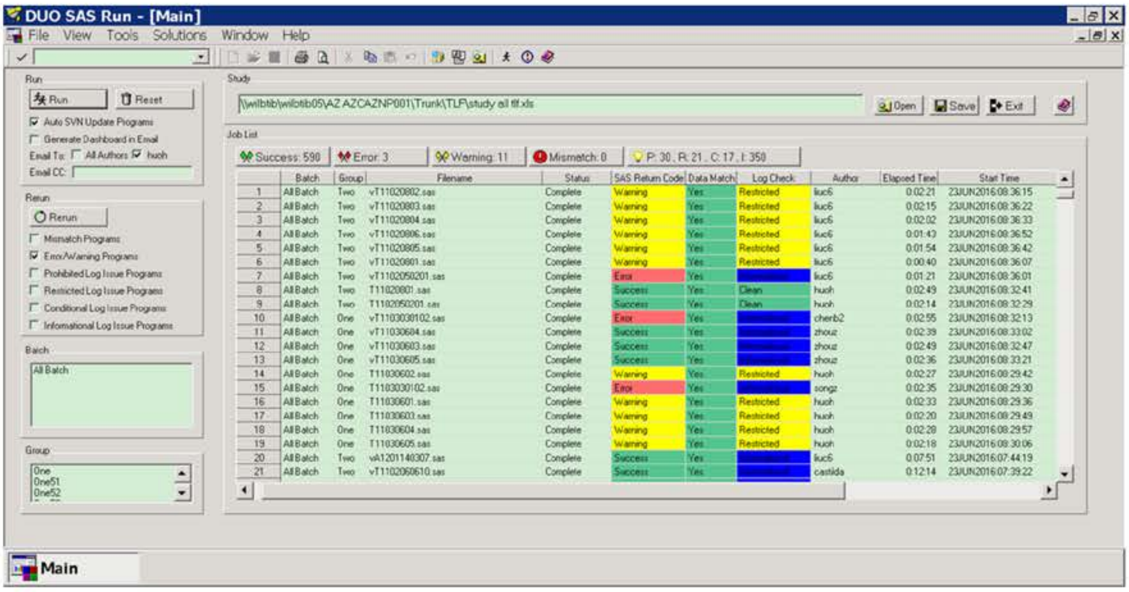
Instead of being assigned by a central job scheduler, the tasks are positively claimed by idle computers. Task pools are accessible for all of the computers in the network. The pool for production side is built when the directory is created by the first computer involved in. But for the QV pool, tasks are added progressively as soon as the production program execution is finished. Those shared pools may lead to conflicts when checking out the same task by multiple computers. SAS/SHARE, which facilitates multi-user management, can be one of the solutions. Since we haven't purchased it, we generate a physical file for each task to indicate its three execution status: “READY", “CLAIMED” and “COMPLETED”, when the task list is created.

Workflows of tasks from being claimed to completed are similar for both sides. The only difference is that for production side, execution order based on the priority level must be taken care of, while it is not a matter for the QV side. In production side, tasks with lower priority level will not start to execute until the last one of higher level is finished. Therefore, each time when idle computers check out tasks from the pool, those with lowest level will be claimed first. In this scheduling mechanism we adhere to first come first served (FCFS) principle. When two or more users try to pick up the same task from the pool, only the first user will succeed. The fulfillment of checking out is marked by updating of status from “READY" to “CLAIMED”, which is achieved by renaming of the physical file. Only one computer will complete the progress to take the ownership for execution. As soon as the execution of the task for production side is finished, the status will be updated to “COMPLETED” and it will be moved to QV pool. The task lifecycle will be performed repeatedly until the last one is finished.

The workflows of performing each program for both sides are similar, but the task pools are different. The tasks for production pool are filled in when the directory is created, while the tasks in the QV pool are added as soon as it is completed for the production side.

Work Flow

This tool was implemented using SAS/AF and SYSTASK statement. The general approach is to use SAS/AF to implement the GUI, by calling the %MAIN macro in RSUBMIT. This macro could use SYSTASK statement to launch a number of SAS sessions to run in parallel. The following is the GUI display.



☑ Auto SVN Update Programs
☐ Generate Dashboard in Email
Email To: ☐ All Authors ☑ huoh
Email CC:

☐ Mismatch Programs
☑ Error/Warning Programs
☐ Prohibited Log Issue Programs
☐ Restricted Log Issue Programs
☐ Conditional Log Issue Programs
☐ Informational Log Issue Programs

Batch: All Batch

Group: One
One51
One52

Study: \\wilbtib\wilbtib05\AZ AZCAZNP001\Trunk\TLF\study all tlf.xls

Success: 590 | Error: 3 | Warning: 11 | Mismatch: 0 | P: 30 , R: 21 , C: 17 , I: 350

	Batch	Group	Filename	Status	SAS Return Code	Data Match	Log Check	Author	Elapsed Time	Start Time
1	AllBatch	Two	vT11020802.sas	Complete	Warning	Yes	Restricted	liuc6	0:02:21	23JUN2016:08:36:15
2	AllBatch	Two	vT11020803.sas	Complete	Warning	Yes	Restricted	liuc6	0:02:15	23JUN2016:08:36:22
3	AllBatch	Two	vT11020804.sas	Complete	Warning	Yes	Restricted	liuc6	0:02:02	23JUN2016:08:36:33
4	AllBatch	Two	vT11020806.sas	Complete	Warning	Yes	Restricted	liuc6	0:01:43	23JUN2016:08:36:52
5	AllBatch	Two	vT11020805.sas	Complete	Warning	Yes	Restricted	liuc6	0:01:54	23JUN2016:08:36:42
6	AllBatch	Two	vT11020801.sas	Complete	Warning	Yes	Restricted	liuc6	0:00:40	23JUN2016:08:36:07
7	AllBatch	Two	vT1102050201.sas	Complete	Error	Yes	[illegible]	liuc6	0:01:21	23JUN2016:08:36:01
8	AllBatch	Two	T11020801.sas	Complete	Success	Yes	Clean	huoh	0:02:49	23JUN2016:08:32:41
9	AllBatch	Two	T1102050201.sas	Complete	Success	Yes	Clean	huoh	0:02:14	23JUN2016:08:32:29
10	AllBatch	One	vT1103030102.sas	Complete	Error	Yes	[illegible]	chenb2	0:02:55	23JUN2016:08:32:13
11	AllBatch	One	vT11030604.sas	Complete	Success	Yes	[illegible]	zhouz	0:02:39	23JUN2016:08:33:02
12	AllBatch	One	vT11030603.sas	Complete	Success	Yes	[illegible]	zhouz	0:02:49	23JUN2016:08:32:47
13	AllBatch	One	vT11030605.sas	Complete	Success	Yes	[illegible]	zhouz	0:02:36	23JUN2016:08:33:21
14	AllBatch	One	T11030602.sas	Complete	Warning	Yes	Restricted	huoh	0:02:27	23JUN2016:08:29:42
15	AllBatch	One	T1103030102.sas	Complete	Error	Yes	[illegible]	songz	0:02:35	23JUN2016:08:29:30
16	AllBatch	One	T11030601.sas	Complete	Warning	Yes	Restricted	huoh	0:02:33	23JUN2016:08:29:36
17	AllBatch	One	T11030603.sas	Complete	Warning	Yes	Restricted	huoh	0:02:20	23JUN2016:08:29:49
18	AllBatch	One	T11030604.sas	Complete	Warning	Yes	Restricted	huoh	0:02:28	23JUN2016:08:29:57
19	AllBatch	One	T11030605.sas	Complete	Warning	Yes	Restricted	huoh	0:02:18	23JUN2016:08:30:06
20	AllBatch	Two	vA1201140307.sas	Complete	Success	Yes	[illegible]	liuc6	0:07:51	23JUN2016:07:44:19
21	AllBatch	Two	vT1102060610.sas	Complete	Success	Yes	[illegible]	castida	0:12:14	23JUN2016:07:39:22

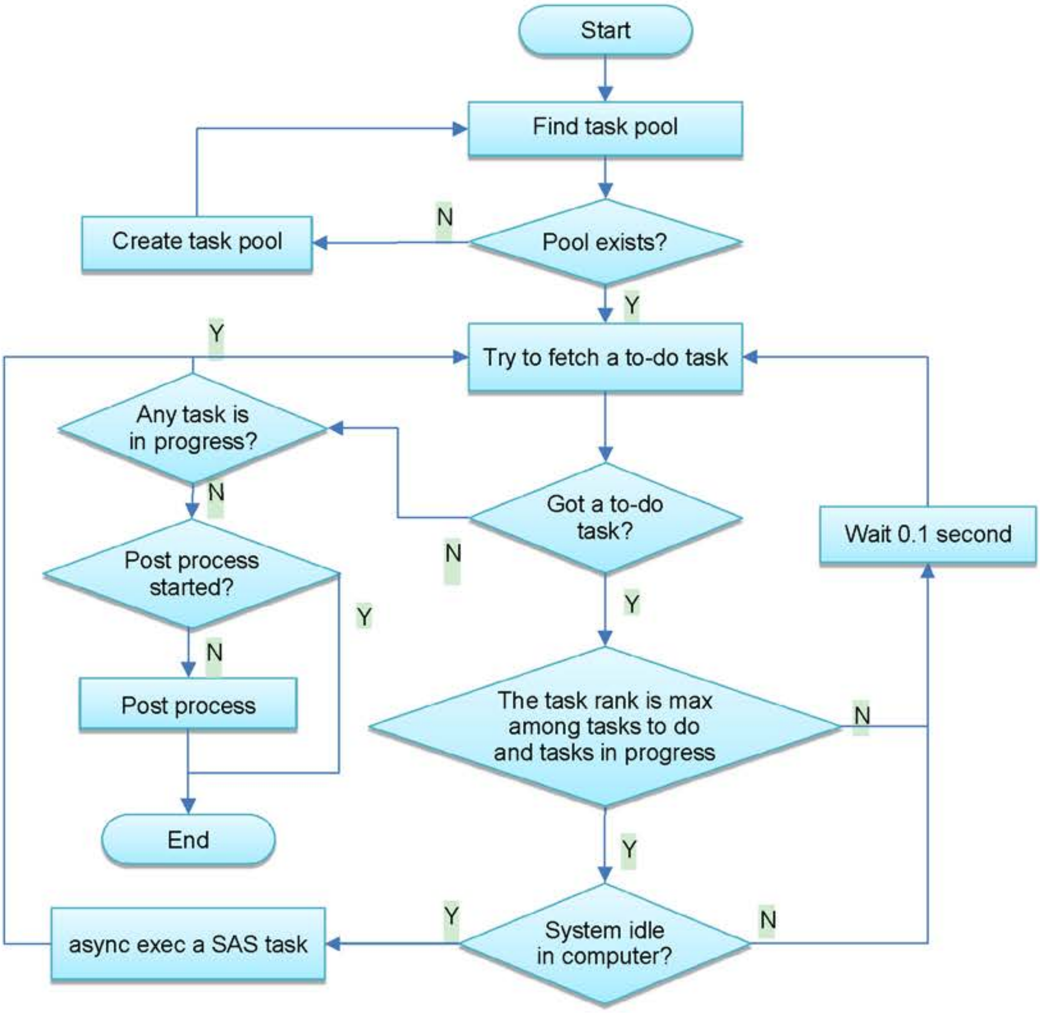
1. Using SAS/AF to develop GUI

Using CONNECTWAIT=NO and RSUBMIT statement, user could get the control of GUI without waiting for the completion of a running program and get the latest status of all programs. SAS/AF is a classic visual programming tool. There is a specific introduction in SAS help. SAS/AF is just one choice to develop GUI. We also could use SAS %WINDOW and %DISPLAY to achieve a real-time interaction. Due to the complexity of the macros and previous papers (Alden 2000, Mace 2002), we will not discuss this part in this paper. We chose SAS/AF since it has a stronger interactions than %WINDOW and %DISPLAY. You can also use color marks to identify the different types (Success, Error, Warning, the Log issue, etc.). By clicking on the program name of GUI, the tool can automatically open the corresponding SAS program and the corresponding Log. That makes debugging and fixing the programs much more easily. The table viewer controls also bring filter selection function and that makes it easy to find what you need from numerous programs.

2. General overview of the %MAIN macro

When running this macro, user should create a list of programs which contains all programs needed in parallel batch submission. There are many methods to assemble this. We recommend using excel file as a best practice.

Brief flow chart of the macro is as follow:



%MAIN can start a number of SAS sessions to run SAS programs in parallel using SYSTASK statement. At the same time, it can automatically examine the running status of each SAS session. When a task is completed, a new SAS session will be assigned to run. In this case, we do not utilize the existing session to start another program run in order to preserve independent execution environment for the programs. This is because during an execution, a program may modify system options, global macro variables, temporary data sets, formats, templates, etc. and may interfere with the execution of next program run. The macro automatically looks up the program owner and the executer's information and sends a summary email to them when a problem occurs.

```
%MAIN(
    jobFileName = StudyName_Joblist.xls,
    jobFilePath = %str(&xlsPathName)
);
```

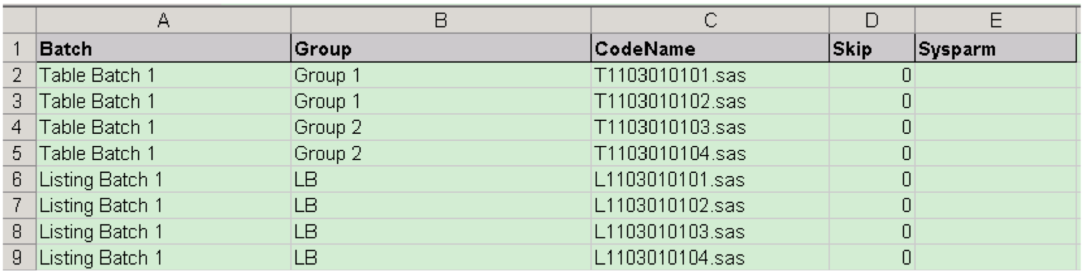
Parameter explanations:

`jobFileName` - Excel file contains program list to batch run

`jobFilePath` - The excel file path

3. Program execution level definition

The hierarchy of the programs must be defined in a parallel batch submission. Using Excel to define the batch and group level is easy to read and maintain.



	A	B	C	D	E
1	**Batch**	**Group**	**CodeName**	**Skip**	**Sysparm**
2	Table Batch 1	Group 1	T1103010101.sas	0	
3	Table Batch 1	Group 1	T1103010102.sas	0	
4	Table Batch 1	Group 2	T1103010103.sas	0	
5	Table Batch 1	Group 2	T1103010104.sas	0	
6	Listing Batch 1	LB	L1103010101.sas	0	
7	Listing Batch 1	LB	L1103010102.sas	0	
8	Listing Batch 1	LB	L1103010103.sas	0	
9	Listing Batch 1	LB	L1103010104.sas	0	

We could use batch and group to define a level. For example, the ‘Group 1’ of the ‘Table Batch1’ tables will be in a same level. There is no sequence within one level so that they could be parallel batch submitted. However, among different levels, there is a sequence, and different levels must be batch submitted in the right sequence.

4. Submit a program

Each program will be submitted by %runOneJob macro and SYSTASK statement. The structure is as follows:

```
%macro runOneJob(
    jobName=
   ,flag=
   ,type=
) ;
   %global &type.done&flag.;
   systask command "'c:\progra~1\sas9~1.2\sasfou~1\9.2\sas.exe'
      -autoexec  ""&G_fullpath.AUTOEXEC.SAS""
      -sysin ""&G_fullpath.&jobName..sas""
      -log ""&G_fullpath.&jobName..log""
      -print ""&G_fullpath.&jobName..lst""
      &initStatement.  &termStatement.
      -noterminal -rsasuser -nosplash
      " nowait status=&type.done&flag.
      taskname="&type.&flag.";
   %if &SYSRC >0 %then %do;
      %put Batch SAS job &G_fullpath.&jobName..sas start failed: SYSRC=&SYSRC..;
   %end;
   %global startDT&flag;
   %let startDT&flag=%sysfunc(datetime());
%mend  runOneJob;
```

Key points:

- SYSTASK asynchronous mode

SYSTASK allows user to execute operating system-specific commands from within user’s SAS session or application. SYSTASK runs these commands as asynchronous tasks, which means that these tasks execute independently of all other tasks that are currently running. Asynchronous tasks run in the background, so user can perform additional tasks while the asynchronous task is still running. SYSTASK asynchronous mode is the cornerstone of our tool.

- SYSTASK NOWAIT option

Determines whether SYSTASK COMMAND suspends execution of the current SAS session until the task has completed. For tasks that are started with the NOWAIT argument, we use the WAITFOR statement to suspend execution of the SAS session until the task has finished.

- RSASUSER ensure the SAS session independent

This option specifies to open the SASUSER library for read access. It is useful since then all SAS sessions will use a single SASUSER library without conflict risk.

- The method of concurrently write dataset

When programs are run in parallel, concurrent conflicts in writing a same dataset can happen if user does not have a permit to use the SAS/SHARE. We could use INITSTMT statement to solve this problem. We use INITSTMT statement to create a library “Here” for each session. This library points to a solely physical path (different programs have different paths). When all program executions complete, one could set all datasets together.

```
-initstmt '||"'"libname here "'" ||"d:\&&&&jobNameOfLevel&&L&level..N&curNumofLevel.."||"'";"'"
```

5. Sending emails

When batch submission completes and results are available, this tool sends emails to authors and program executor.

CONCLUSION

Benefits

Our old version tool invoked multiple SAS session by utilizing CPU in a time-division-multiplex mode in a single CPU. The efficiency is enhanced significantly. The computation efficiency is still limited by its own CPU and memory limitation. When it reaches the limit of physical capacity, the system performance will be affected negatively. To maximize the power of the tool, tasks are dispatched to different CPUs/cores in grid computing system. The benefit of this tool are stated as below.

(1) **Cost effective**: As central server and distributed virtual computer is common in pharmaceutical industry. Centralizing arrays of virtual computers to fulfill a set of SAS program execution is easy to achieve in this circumstances.

(2) **Utilization of idle resource**: By utilizing the idle resources especially in business hours to reduce the time for waiting makes work more productive.

(3) **Modularity**: Each of the tasks can be work independently. If one of the processor crashes, others will not be affected. The failure job can be picked up by other computers.

(4) **Resource-adaptive**: Tasks was claimed by the execution computer based on its own resource instead of being assigned by a dispatcher.

(5) **Flexible**: The number of computers in the grid is not pre-defined. Any idle computers available can be joined any time. If it is not available, it can be removed.

Next Step

Currently, we still need to ask people to log in to acquire idle computers. In the future, we plan to use SAS/IT to enhance this tool to support Browser/Server mode. By optimization the IT resources across enterprise can facilitate to maximize the use of the tool.

REFERENCES

Alden, Kay. 2000 "SAS' Best Kept Secret: Macro Windows® for Applications Development" Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference, paper 76.

Cawley, Jim, Prescod Jillian 2010 "Need Reporting Deliverables the Painless and Easy Way? A Batch Submit Macro of Course!" Proceedings of the PharmaSUG 2010 Conference, paper CC10.

Chen, Ling Y., Gilibet Steven A. 2002 "Run All Your SAS(R) Programs in One Program Automatically" Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference, paper 105.

Conover, William 2011 "BAT Files: Run all Your Programs with One Click in PC SAS" Proceedings of the PharmaSUG 2011 Conference, paper PO01.

Mace, Michael A. 2002 "%WINDOW: You Can Talk to the Users, and They Can Talk Back" Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference, paper 192.

Murphy, Howard. 2007 "Changing Data Set Variables into Macro Variables" Proceedings of the 2007 SAS Global Forum, paper 050.

SAS Institute Inc. 2013 *SAS® 9.2 Companion for Windows* Cary, NC: SAS Institute Inc.

Shu, Haibin 2006 "Highly Effective Batch Processing" Proceedings of the PharmaSUG 2006 Conference, paper AD09.

Sun, Helen, Wong Cindy 2010 "A Macro to Create a Batch Submit SAS Program" Proceedings of the 2010 SAS Global Forum, paper 092.

RECOMMENDED READING

Cogswell, Denis. 2005 "More Than Batch – A Production SAS® Framework" Proceedings of the Thirtieth Annual SAS Users Group International Conference, paper 21.

Furdal, Stanislaw. 2008 "Quick Windows Batches to Control SAS® Programs Running Under Windows and UNIX" Proceedings of the SAS Global Forum 2008, paper 17.

Andrew E. Hansen. 2013 "A Macro to Batch Submit a List of Programs with Real Time Feedback" PharmaSUG 2013, paper AD15

ACKNOLEDGEMENTS

The author would like to thank Ping-Chung Chang and Xianyi Kong for their insightful comments in reviewing an earlier publication of this paper.

DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of PPD.

CONTACT INFORMATION

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

Name: Huashan Huo
Enterprise: PPD LLC.
Address: 25th Floor, Raffles City Beijing Office
Tower No.1 Dongzhimen South Street,
Dongcheng District, Beijing 100007, China
Work Phone: +86 10-61846092
Fax: +86 10-61846099
E-mail: Huashan.Huo@ppdi.com
Web: www.ppdi.com

Name: Zhongyu Li
Enterprise: PPD LLC.
Address: 25th Floor, Raffles City Beijing Office
Tower No.1 Dongzhimen South Street,
Dongcheng District, Beijing 100007, China
Work Phone: +86 10-61846085
Fax: +86 10-61846099
E-mail: Zhongyu.Li@ppdi.com
Web: www.ppdi.com

Name: Lu Zhang
Enterprise: PPD LLC.
Address: 25th Floor, Raffles City Beijing Office
Tower No.1 Dongzhimen South Street,
Dongcheng District, Beijing 100007, China
Work Phone: +86 10-61846094
Fax: +86 10-61846099
E-mail: Lu.Zhang18@ppdi.com
Web: www.ppdi.com