Paper 220-30

Dynamic Load Balancing for SAS® ETL Studio 9.1

Frank Pecjak, SRA International Inc, Fairfax, VA
David Schwartz, SRA International Inc, Fairfax, VA

ABSTRACT

SAS ETL Studio 9.1 provides you a convenient, workflow based interface, to efficiently move data across disparate data sources. The flexibility of the interface and the ease with which you can add code make ETL Studio a robust ETL solution that can solve a wide variety of data movement issues.

As the size of a dataset increases, the scalability of a particular load task becomes a critical issue. Typical approaches to resolving scalability issues include breaking up datasets into multiple jobs, parallelizing tasks across multiple servers and potentially running the job on more powerful machines. The goal of this paper is to identify the advantages and disadvantages of the tuning options available through ETL Studio.

INTRODUCTION

SAS ETL Studio 9.1 provides you a convenient, workflow based interface, to efficiently move data across disparate data sources. The flexibility of the interface and the ease with which you can add code make ETL Studio a robust ETL solution that can solve a wide variety of data movement issues.

As the size of a dataset increases, the scalability of a particular load task becomes a critical issue. Typical approaches to resolving scalability issues include breaking up datasets into multiple jobs, parallelizing tasks across multiple servers and potentially running the job on more powerful machines. ETL Studio provides the capability to tune jobs in each of these ways and also provides a tool, through the Metadata Server environment, to help choose the right environment dynamically.

The goal of this paper is to identify the advantages and disadvantages of the tuning options available through ETL Studio. In addition, we propose an alternate approach which utilizes the robust configuration ability within the workflow in order to potentially realize the advantages from all tuning options simultaneously. While our approach has been specifically designed for use in ETL Studio, we are confident that it can be easily extended to any multi-step SAS process residing in a multi-node environment.

This paper assumes a slight familiarity with the SAS ETL Studio front-end and tasks. A solid understanding of the SAS 9 Metadata Server and it's components as well as the usage of SAS\Connect will also be helpful when reviewing our solution. Finally, our solution utilizes macros to evaluate system resources; therefore some familiarity with the use of macros in SAS code would be beneficial.

CURRENT STATE OF LOAD BALANCING IN SAS ETL STUDIO 9.1

Like any other Extract, Transform and Load (ETL) tool, jobs created in ETL Studio can quickly become complex to manage and resource intensive as the amount of data and requirements for individual transformations rise. Tuning these jobs can be accomplished in two ways. First, utilizing the Load Balancing Server capabilities within the Workspace Server and second by specifying a server to run individual processes within a job.

USING THE LOAD BALANCING SERVER

By working through the Workspace Server, a SAS ETL job utilizes the load balancing algorithms that come with the SAS Object Spawner. A job submitted through the Object Spawner is routed to the least encumbered server. This process happens behind the scenes with no input from the submitter. By submitting the job to the appropriate Object Spawner, the spawner will then route the job to the Workspace Server that is the least encumbered. The disadvantage of this solution is that it routes the entire job to a single server. For a single ETL job, which can perform many resource intensive tasks such as sort, merge, or aggregate; running all the steps of the job on a single server may not be an optimal use of resources. Even if the steps are synchronous in nature, it may be worthwhile to reevaluate before each large step if the server where the job is running is still the best choice and if not, submit that step to a more optimal server.

SPECIFYING THE DATA SERVER

As part of the configuration of a step, a developer can choose to use the default server or to specify a server where the step is to run. This specification will cause the job to open a SAS/Connect session with the appropriate server and the resulting dataset is transferred back to the parent process. A SAS/Connect session is not subject to the load balancing as described above. This option does offer more fine tuned control of where resource intensive steps are run, and works well to ensure that branched logic can be run asynchronously. However, the server specification is “hard-coded” into the step and does not take into account the available resources on a machine. In the end, this manual intervention could potentially slow an ETL job down rather than make it more efficient.

DYNAMIC LOAD BALANCING

Obviously, the most complete solution for balancing jobs is to combine the dynamic balancing capabilities of the Load Balancer with the granular control available in ETL Studio. Fortunately, the code generated by ETL processes is available for edit so we have the ability to

specify by process the server wrting the SQL\Connect SIGNON statement. By adding a custom code process before a large transform, we can add mimic the capabilities of a Load Balancing Server to choose an appropriate resource to submit the process.

THE %CHOOSESVR MACRO

We offer the %CHOOSESVR macro as a means of providing the load balancing capabilities to the individual processes within a job. The macro will query the operating system for each available machine to assess the available resources. It will then return the machine best able to handle another SAS process.

The %CHOOSESVR macro will accept the number of servers in your environment as an argument. A pre-requisite for using the macro is that you have a list of the servers in your environment defined. This could be done in several ways, for example a dataset that specifies multiple arguments for each machine: the name (or IP address), port, userid, password and operating system (Windows or UNIX) for each available SAS server. In example provided here, we are only working with on operating system (UNIX) and so the Macro is written with the server list defined as macro variables. Either approach works fine as long as the macro can read in all the information it needs for each server environment. It is also assumed here that the SAS servers specified must are registered with the Metadata Server, and have an Object Spawner with a SAS\Connect port specified. It would be possible to modify the macro to us a script for establishing connections, but that topic will not be addressed here.

For each server specified, the macro performs a SIGNON and uses operating system specific commands to assess the resources available on the machine. Our initial prototype counts the number of SAS processes (minus one for the counting process). In environments where the servers are dedicated to SAS work, you could easily add ‘grep SAS’ to the system command and pull in only processes that were initiated by the SAS command. The machine with the smallest number of processes running has its name placed in a global macro variable. The macro can be rerun before every process in the specified job, and each process can be run on the least used server at the moment that a new ETL node begins.

Future iterations of the macro will allow for a more robust assessment of the operating resources available. For example, on a UNIX system, a ‘vmstat’ command can be performed to analyze the available memory, disk and processor utilization for the machine, regardless of what the existing SAS processes are. The final result for these commands can be compared to determine the least encumbered like the prototype or the data can be used in a more robust analysis, such as a regression, to determine the best machine to use. The point is that there are many ways to assess which is the best machine to use, however, any algorithm must balance robust assessment with fast response. The routing algorithm should not unduly burden the overall ETL job.

The macro begins with the definition of some macro variables needed for execution:

```
%let machine1 = 'miner.sra.com';
%let machine2 = 'miner2.sra.com';
%let machine3 = 'miner3.sra.com';
```

You could also define macro variables here for port, UserID, or Password, etc as long as the format for machine name is followed (e.g., machine1_port). These additional variables could then be added to the iterative do-loop to affect authentication, non-standard ports for the spawner programs, or other SAS/Connect options. As noted earlier, this could also be handled by a SAS data set, in either case this information need only be defined when the macro is defined. Once this the machine names are set up, the macro reads then in and establishes a brief query of each server, and in this case reads in the results of a ‘ps –f’ command.

```
%macro choosesvr(srvnum=);
    %do i = 1 %to &srvnum;
           %let local = &&machine&i;
           options remote=local;
           options comamid=TCP;
           signon local noscript;
           rsubmit;
                  data serv_dat;
                         call system('ps -f > /export/home/sas/srv_out.txt');
                         wait=sleep(30,1);
                         infile = '/export/home/sas/srv_out.txt';
                         input UID PID PPID C STIME TTY TIME CMD;
                  run;
                  proc download in=serv_dat out=serv_dat&i;
                  run;
           endrsubmit;
           signoff rhost;
           data all_data;
                  set all_data serv_dat&i;
                  machine_name = machine&i
```

```
        run;
    %end;
    proc summary data= all_data;
        class machine_name;
        output out= all_data_sum  n=pid;
    run;

    data min_pid;
        set all_data_sum;
            pid = pid-1;
            retain maxpid;
            maxpid = max(maxpid, pid);
            if maxpid > pid then call symput("choosehoststring",machine_name);
    run;
%mend;

%choosesrv(srvnum=3);
```

INTEGRATING %CHOOSESVR INTO ETL STUDIO JOBS

Integrating the macro into an ETL Studio job is a two step process. First, a specific machine must be specified in the Process tab(See Figure 1 below). Note that the machine specified is may not be the machine that runs the process. In fact, it need not be a machine that is listed in the available machines dataset. By specifying the machine, ETL Studio inserts the necessary SAS\Connect information to submit the complete process code. While this step can also be performed manually, this ensures all code is captured and the ETL global macros are populated appropriately.

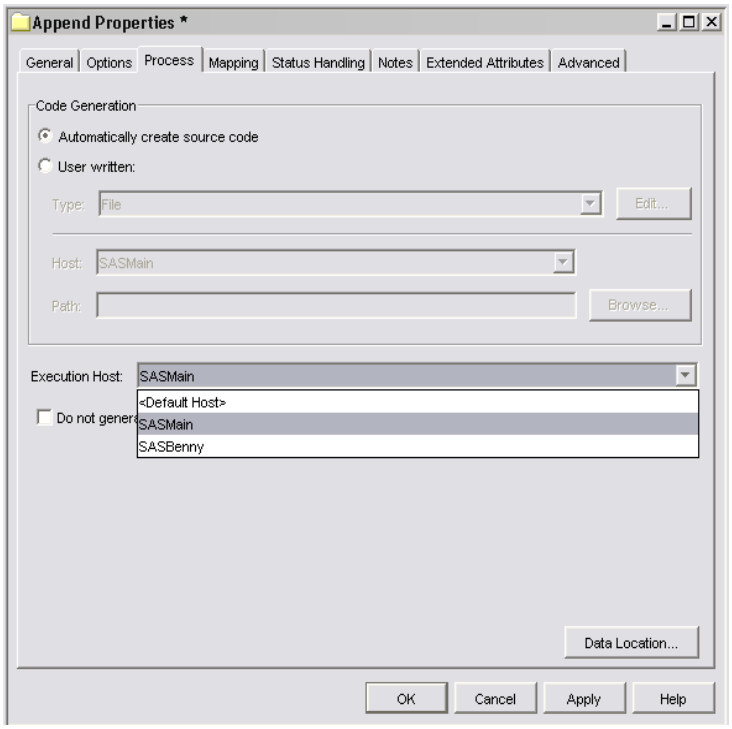

Figure 1 - Host Specification

Looking at the code created by ETL Studio, as shown below, a macro variable is created that is passed the SIGNON statement. The %CHOOSESVR macro will populate this variable.

```
options comamid=TCP;
%let local=%choosehoststring;
signon local  noscript;
```

Unfortunately, these edits have to be done manually. To ensure that the code does not get overwritten, you should make any other edits to this process first, generate the source code, then reedit the process and check the Do Not Generate Code for This Process checkbox. Also,

we must remember to uncheck the option and reapply the changes if the process were to be changed.

The final edit is to add the macro execution script before every process to load balance. This addition is done via a User Defined Code node(see Figure 2 below).

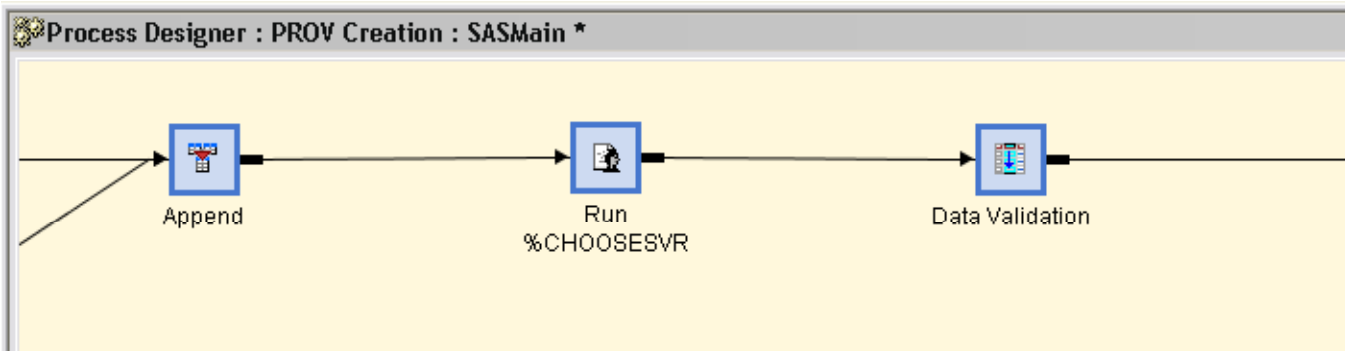

Figure 2 - Apply User Defined Code Process

The first node defined must include the macro code before it runs the process. Once included just the call can be put into subsequent code nodes. A sample of the code added to the User Defined node is shown below:

```
%include 'C:\project\choosesvr.sas';
%choosesrv(srvnum=3);
```

BENEFITS OF USING DYNAMIC BALANCING

The goal of these additions to the ETL job is to more effectively manage resources. In an environment with two available servers, the benefits our macro provides would probably not realize an advantage over the SAS Load Balancer. The goal of the macro is to effectively utilize the total resources available to run SAS jobs. While the macro will work well in the two server case, its true benefit comes through with three or more servers, as it extends beyond the capabilities of the Load Balancer and standard ETL design functionality. Another advantage of the macro in managing machine resources is that, as future versions look at more robust measures of system resources, the macro will have a more complete view of the system than the Load Balancer so it should make better choices for routing processes.

In addition to the number of servers available to %CHOOSESVR, the complexity of the ETL job plays a role in the effectiveness of the macro. A job that runs processes serially will realize some benefit. In this case, the macro will utilize the resources and ensure that each node will run on the machine most able to run the process. This ensures that the overall job will run at close to constant times over multiple executions. When looking at parallel processes, %CHOOSESVR will again ensure that each node runs on the most effective machine, which will have the effect of lowering the overall execution time versus what may be accomplished with the out of the box load balancing techniques. Some of these benefits can be realized using the process definition to split the node execution, but because these machines are hard-coded, there is potential that a process runs on a machine that is already burdened.

One disadvantage of our current design is that %CHOOSESVR can potentially choose the same server for two processes that are designed to run in parallel. That is, the macro does not take into account where other processes in the job are being routed. This problem can be solved by utilizing the technique that is already used by ETL Studio, identifying a different set of servers for each concurrent process. That is, we can create a separate server list of each potential concurrent process and separate arguments to each %CHOOSESVR execution. For example, if there are a maximum of three concurrent processes for a particular job, then three separate lists of servers can be created to ensure a minimum amount of overlap. Obviously, this solution is limited by the number of servers available for load balancing.

CONCLUSION

The approach explored in this paper is the result of exploring the ETL studio product. Though we find ETL studio very useful ‘out of the box’, one of its strengths is how easy it is to extend. We found its integration with the SAS9 metadata environment and load balancing capability very useful. Our experience applying ETL to VLDB environments lead us to explore a more robust approach to load balancing. What we have demonstrated here is just one approach and a rather simple one using only on OS. The important point, though, is how easy it was to accomplish. Future iterations of this approach will introduce different operating systems, role based authentication, and the script files. What we hope we have demonstrated here is how rich the ETL studio environment can be made through custom extensions.

CONTACT INFORMATION

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

Frank Pecjak
SRA International Inc
4300 Fair Lakes Court
Fairfax VA 22033
Email: frank_pecjak@sra.com

David Schwartz
SRA International Inc
4300 Fair Lakes Court
Fairfax VA 22033
Email: david_schwartz@sra.com