Paper 212-31

Grid Computing with SAS® - A Developer's Perspective

Greg McLean, Statistics Canada, Ottawa, Ontario, Canada

ABSTRACT

Over the past few years there has been a great deal of discussion and hype over the concept of Grid Computing. It seems that many of us in the Information Technology field are now convinced of the major benefits:

- Ability to reduce overall processing time
- Ability to leverage existing hardware

However, a big question still remains on a lot of people's minds. How does one actually proceed to implement such a strategy using SAS®? Using various components of the SAS software, in particular SAS\CONNECT®, we have the ability (and have had the ability for the past decade) to implement a Grid Computing solution with a somewhat simple amount of effort. Of course some changes will be required to existing SAS programs and/or data.

This paper will focus on the concepts and ideas that the developer will need to address in order to successfully implement a Grid Computing strategy using SAS. A basic discussion of SAS\CONNECT will also be made. The intended audience for this paper is SAS developers with a medium to advanced knowledge of SAS.

INTRODUCTION

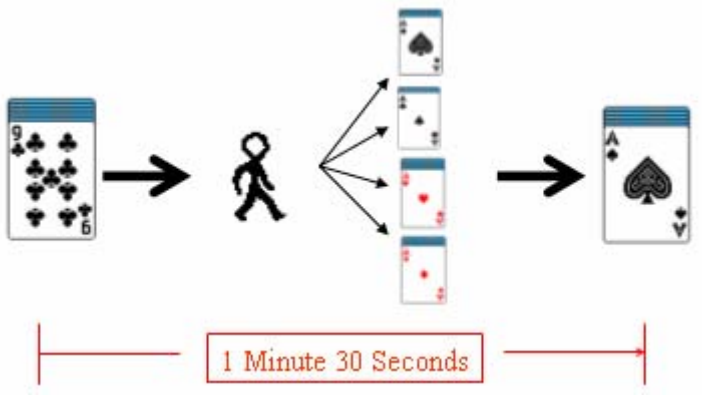
With today's declining budgets, higher volumes of data and reduced available processing time one must find ways to increase efficiency while keeping costs down. A grid computing solution can reduce overall elapsed time while leveraging existing hardware and software. Grid computing allows the processing of data to run in parallel on remote homogeneous or heterogeneous machines.

This paper will focus on the issues that the SAS Developer must address in order to implement a successful Grid Computing strategy. There are three main areas that will be discussed:

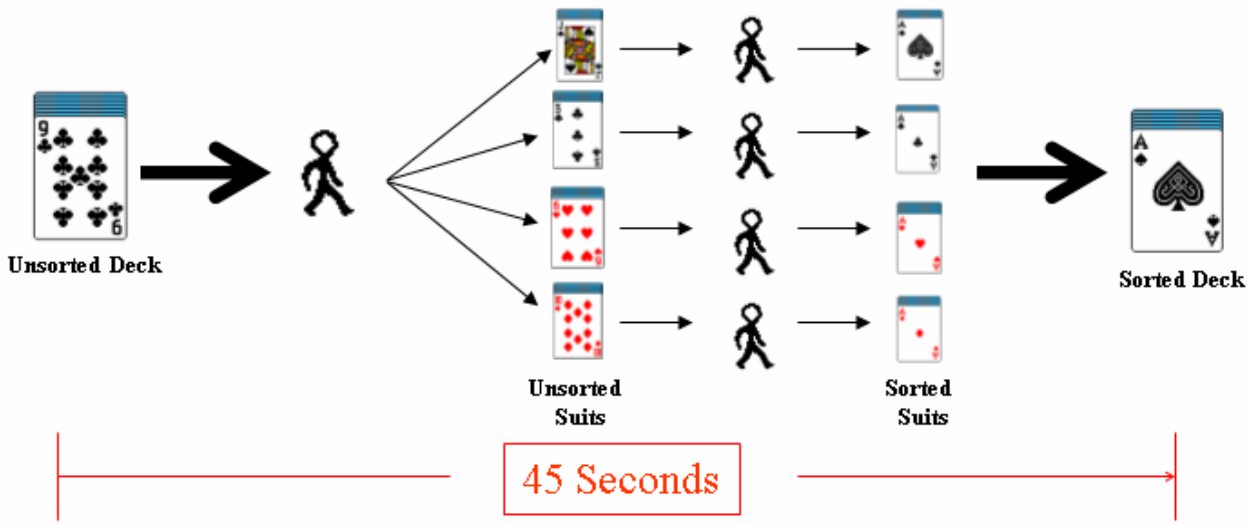
1.) Grid Infrastructure (SAS\CONNECT)
2.) Grid Controller software (SAS\AF®)
3.) SAS Programs \ SAS Data

The primary concept of Grid Computing is the ability to divide a given task (i.e. SAS Program) and /or data (i.e. SAS Data Sets) into subtasks, such that these subtasks can be processed in a parallel fashion.

A great way to illustrate the concept of Grid Computing is by using a common deck of playing cards. Let's assume that we wish to sort the cards from Ace to King within each suit. Using a simple or traditional approach, let us assume that we assign this task to one person.



After some experimentation, it was found that on average it took a person approximately 1 minute, 30 seconds to complete the task. Now let's perform the same task using a Grid Computing strategy. This time we will assign various subtasks to more than one person. We will first need one person to divide the deck into 4 piles (1 pile for each suit). Then we will have 4 people sort one pile each, from Ace to King.

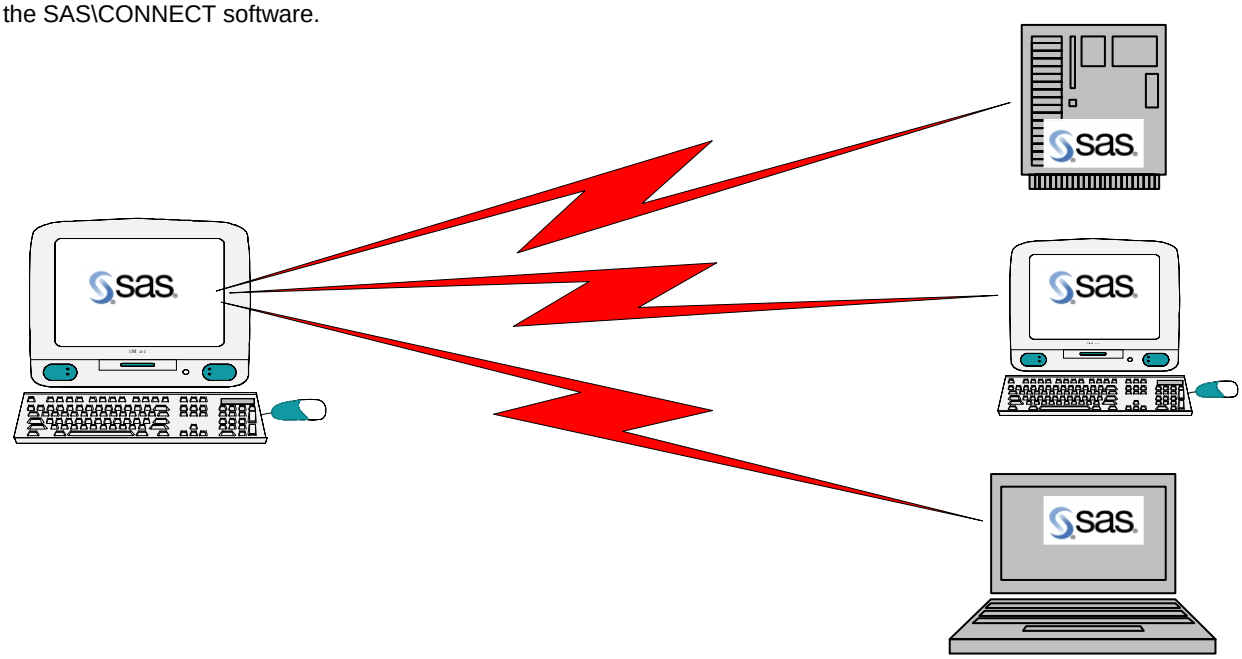


Same task…same result, however, using a parallel processing strategy we are able to perform the same task in a shorter amount of time. On average it took 5 people approximately 45 seconds of elapsed time to complete the task.

The benefit is gained based on the fact that some of the sub-tasks are performed at the same time or in parallel. The only difference in the Information Technology World is that we use computers instead of people in a Grid Computing environment. Now some might argue that more hardware is required to truly achieve these types of gains. True…however, it has been reported by researchers that on average, business computers are only utilizing between: 10% - 20% of their capacity. Therefore, there is no need to acquire new hardware; we simply need to make better utilization of the hardware that we already have.

GRID INFRASTRUCTURE

In a Grid Computing environment, the underlying infrastructure or network is the key to its success. Since Version 6.12, SAS has provided the ability to connect to one or more remote homogeneous or heterogeneous machines using the SAS\CONNECT software.



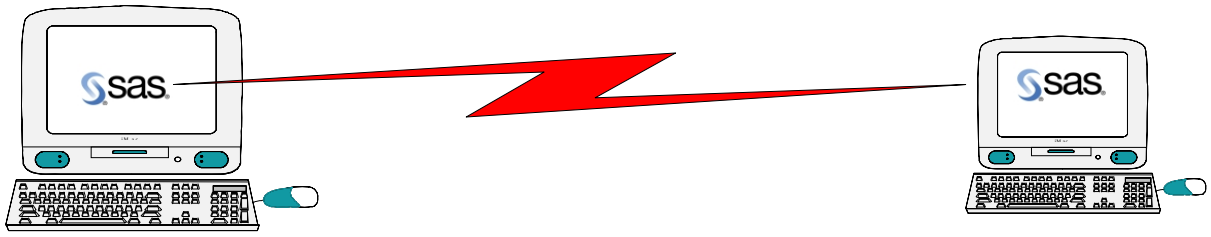
And even more important, particularly for Grid Computing, SAS allows the processing of data to be done in an asynchronous fashion. Because of this, we are able to run SAS programs simultaneously or in parallel on these remotely connected machines. Throughout this presentation we will refer to these remote machines as "**Nodes**". We will refer to the controlling machine as the "**Client**".

Each of these **Nodes** must be setup in a particular fashion to allow the client machine to make the necessary connection. Of course a version of SAS must be installed on all machines to be used on the "Grid" (including the client machine). In some instances, additional preparation will be required. For example, in a PC environment, a SAS "Spawner" service will need to be started and configured on a node. Once all of the machines to be used in the Grid environment have been configured correctly, the SAS Grid infrastructure is theoretically complete.

The concept of client/server is quite simple when using the SAS\CONNECT software. A SAS session is started on the client machine. The client SAS session sends out a request through the network to another machine. A dedicated SAS session is then started on the remote machine. This means that as a user on the client machine, two or more SAS environments are available to submit SAS code (depending on how many remote machines are referenced). One can still locally submit SAS code as usual, however, the option to submit code to one of the remotely connected machines is now available.

The following code example illustrates just how simple it can be to connect from the client SAS session to a **Node** on the SAS Grid. Assuming that a machine called "**STCSERVER**" has been properly configured (SAS, SAS\CONNECT), the following code (submitted from the client machine) should start a dedicated SAS remote session.

```
%let myremote=STCSERVER;
options comamid=tcp remote=myremote;
signon myremote;
```



Client Machine **Server Machine (Node)**

It is not the intention of this presentation to detail the SAS setup requirements of the client and/or server machines. But rather to stress the fact that SAS must be running on all machines involved in the Grid (including SAS\CONNECT software). And in addition, in some circumstances, other preparations may be required.

GRID CONTROLLER SOFTWARE

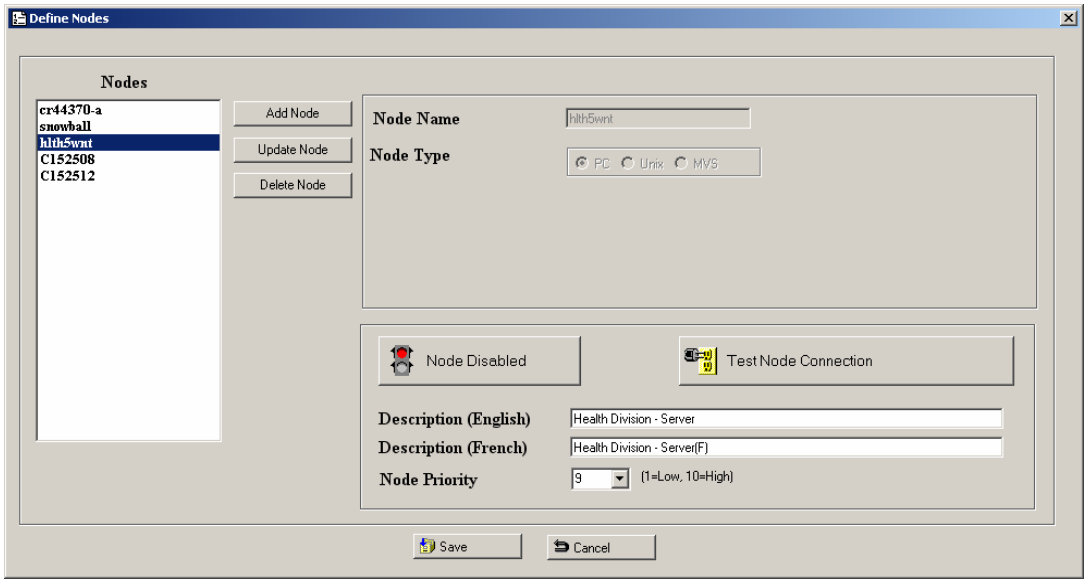
Once the underlying Grid infrastructure is in place, we then need some software that will manage the Grid. In this presentation we will use a SAS\AF prototype developed at Statistics Canada to illustrate the various requirements of the Grid controller. However, it is possible to accomplish the same requirements using other components of the SAS software (for example, SAS Macros).

The Grid Controller Software must manage three different aspects of the Grid:

1.) Nodes
2.) Process Flow (Sub-tasks)
3.) Job submission

NODES

The Grid controller software must know about all nodes that are considered connected to the grid. This may include the node name, location, and description. In some instances a SAS\CONNECT script file will also be required. In the prototype it was decided to add a concept referred to as "**Node Priority**". It is basically a user specified ranking of the power of the server or node (from 1 to 10, where 10 is the most powerful). Therefore at run-time, the Grid controller software can match more complex sub-tasks to the more complex nodes (optimization issue).



In the above example, 5 different nodes have been defined to the prototype. The information about each node is actually stored in a SAS Data Set. Various other utilities were added to the screen to improve functionality and usability. The important point to make is that somehow the node information needs to be stored and available to the Grid controller software.

PROCESS FLOW (SUB-TASKS)

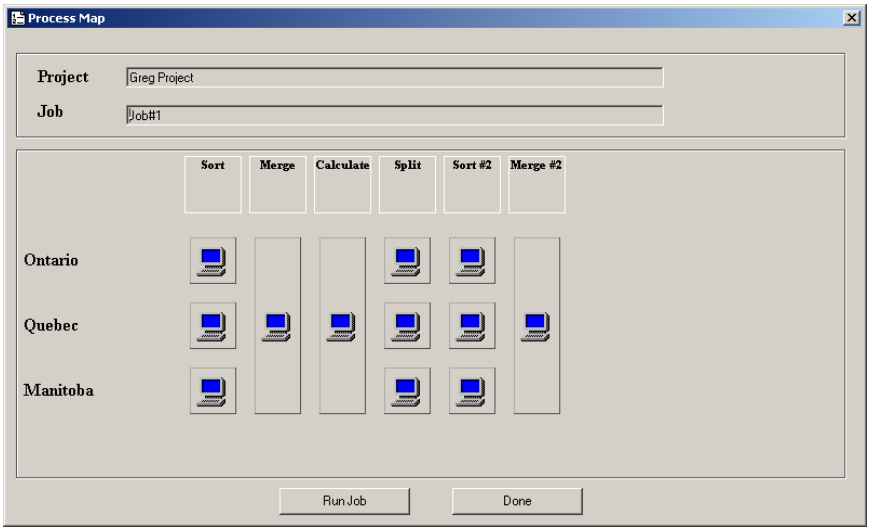
As mentioned in the introduction, the goal of Grid computing is to break down tasks into sub-tasks, such that the sub-tasks can be processed in parallel, thus reducing the overall elapsed processing time. In SAS, when we refer to a task, we are referring to a SAS program. It may contain many data steps, procedures, macros and so on. It is then necessary for the user to identify ways to split the main SAS program (task) into smaller, separate programs (sub-tasks) and to be able to define these sub-tasks to the Grid Controller software.

Example:

Original SAS Program (Task)	Modified SAS Programs (Sub-tasks)
LIBNAME IN '\\Server1\Input'; LIBNAME OUT '\\Server1\Output'; PROC SORT DATA=IN.DATA1; BY KEY; RUN; PROC SORT DATA=IN.DATA2; BY KEY; RUN; DATA OUT.FINAL; MERGE IN.DATA1 IN.DATA2; BY KEY; RUN;	LIBNAME IN '\\Server1\Input'; PROC SORT DATA=IN.DATA1; BY KEY; RUN; LIBNAME IN '\\Server1\Input'; PROC SORT DATA=IN.DATA2; BY KEY; RUN; LIBNAME IN '\\Server1\Input'; LIBNAME OUT '\\Server1\Output'; DATA OUT.FINAL; MERGE IN.DATA1 IN.DATA2; BY KEY; RUN;

In this example, it is possible to create three separate programs from the original SAS program. Each of the three programs could be run on their own separately. However, the order in which these sub-tasks run is important. For example, we must run the first two sub-tasks (sorts) before we can run the third sub-task (merge). And in a Grid Computing environment it is also important to keep in mind that any given sub-task can be run on any connected node. This becomes important when defining directory and/or file paths. See the “Other Considerations” section for more detail.

In the SAS\AF prototype, the user has the ability to create a visual representation of the sub-tasks, including the process flow (icons). Once the visual representation is complete, the user then has the ability to define a separate SAS program for each visual object. In the case of the prototype, this information is stored in a SAS Data Set.

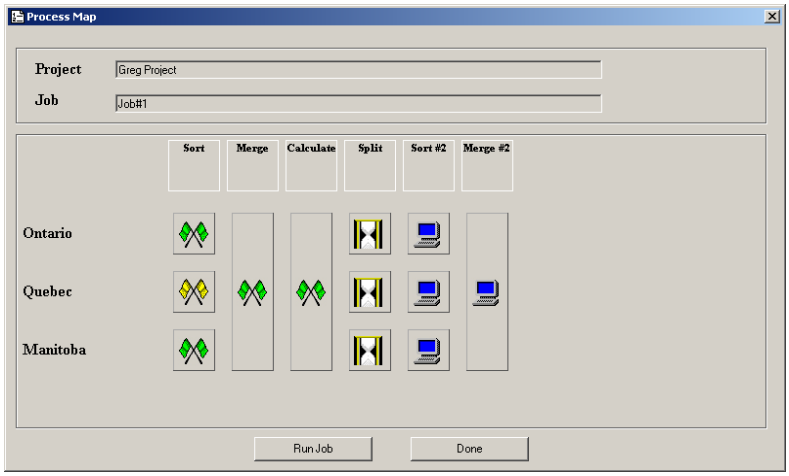


To summarize, sub-tasks in any given column can be run at the same time or in parallel. Sub-tasks in a given column cannot run until their predecessor sub-tasks (the sub-task(s) in the column to the left) have completed. To summarize, the process flow reads from left to right. For example, the sub-task identified in the “Merge” column cannot be run until all sub-tasks in the “Sort” column have completed.

JOB SUBMISSION

Now that the Grid controller software knows about the connected nodes and has the process flow information of the SAS job, one can use the prototype to submit and manage the distribution of each sub-task.

One advantage of using SAS\AF to act as the Grid controller software is that it is more visual. The users can see the status of any given sub-task in real-time:



completed successfully

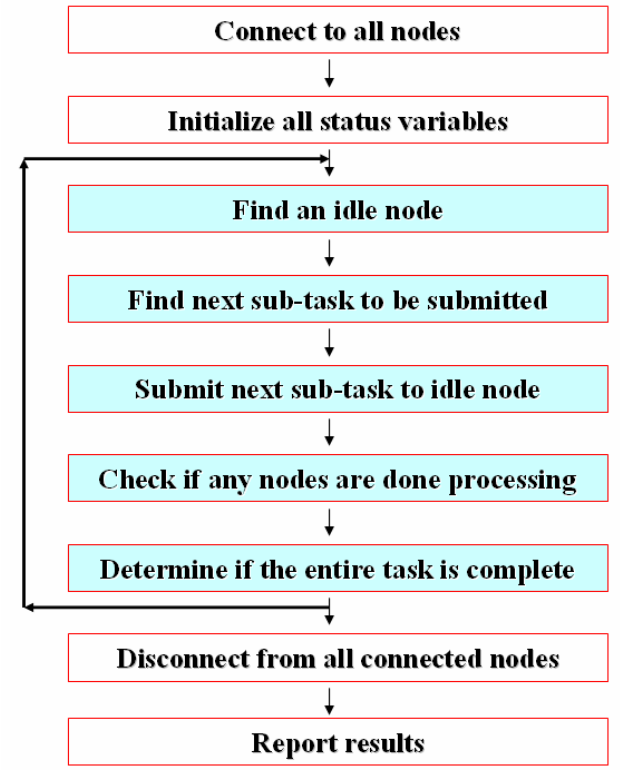
completed with WARNINGS

not complete due to ERRORS

currently running on a node

waiting to be run

Visually it appears to be quite simple; however, internally there is a great deal more going on. Basically the Grid controller software must act as a “Daemon” that continually iterates through some the following steps until either the entire job has completed or a given sub-task has failed.



Connecting to each server can be as simple as illustrated in the “Grid Infrastructure” section. Once the nodes are connected, we then need to setup some status flags to keep track of what is going on in each connected node. A simple and effective approach is to use SAS macro variables.

The middle section, highlighted in blue requires a little more ingenuity. Basically, the software continually checks the status flags, updates system tables, updates status flags, submits sub-tasks and so on.
Once the entire job has completed, various summary statistics could then be presented to the user. For example, total processing time or number of nodes utilized, etc.

Job Summary | Job Details

Project	Job	Submission Number	Status	Elapsed Time	Start Time	End Time	Number of Nodes	Number of Tasks	Submitted by
1	1	7	E	0:00:02	2:29:51 PM	2:29:54 PM	3	12	mclegry

SAS PROGRAMS / SAS DATA

One of the big misconceptions about Grid Computing is that one can simply run existing programs through the system with no change. Unfortunately this is not true. There will almost always be some modifications required to one’s SAS program(s). However, one of the goals of the Grid controller software is to minimize the amount of change required.

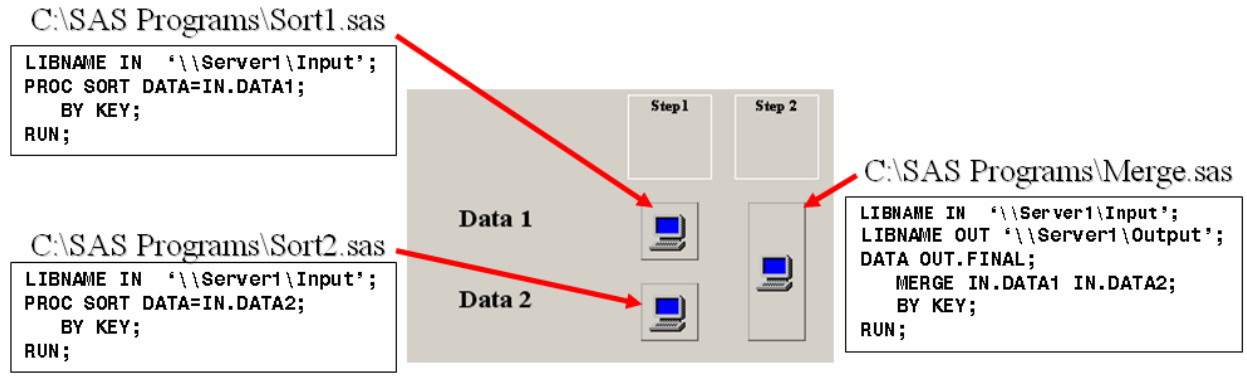
Once the process flow of all sub-tasks has been visually defined, we must attach a separate SAS program to each icon in our process flow map. In this section we will discuss three different ways in which a given SAS program (task)

can be modified to run in a Grid environment using SAS. The three methods to be discussed are:

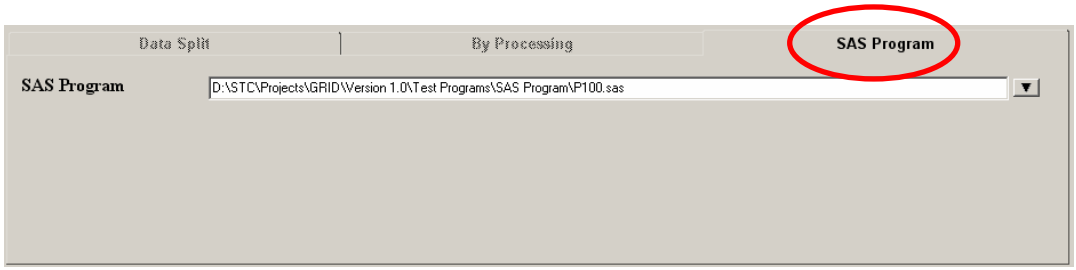
1.) Individual SAS Programs
2.) By Processing Programs
3.) Data Split Programs

INDIVIDUAL SAS PROGRAMS

When creating the first version of the prototype, it was desired to keep the processing as simple as possible. This meant that for each defined icon in the process map, there would be a separate SAS program. The user would simply divide one SAS program into several smaller programs and save them to separate files.



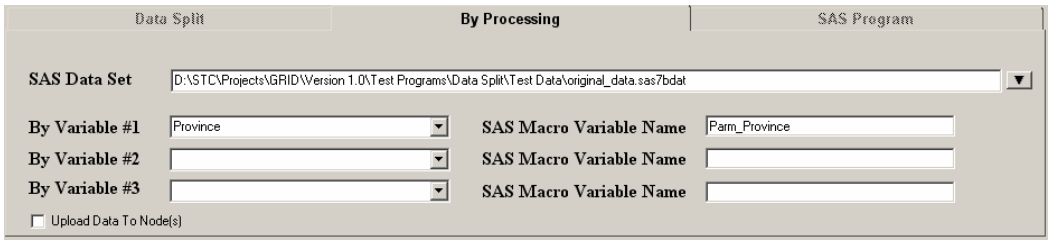
To summarize, for each icon, the Grid controller software would need to know the location and name of the SAS program to run.



BY PROCESSING – SAS PROGRAMS

Although allowing the user to supply a separate SAS program to each icon is useful, it was quickly realized that this would not always be a simple approach. What if a given SAS program was processing using a SAS "BY" variable? For Grid Computing, the obvious thing to do would be to create a separate SAS program for each "BY" variable value. Take for example a program that processes data by Province (Canada). This would mean that the developer would have to create 13 separate SAS programs (which would be almost identical except for the Specified Province) and consequently, 13 icons in a column in the prototype. And what if the BY variable value contained hundreds of possible values. It would not be practical to manually create a separate SAS program for each.

A solution for this problem would be to allow the user to define only one icon, but allow the Grid controller software to dynamically create separate SAS programs when processing. This would mean that the Grid controller software would need to know some additional information besides the name of the SAS program. The Grid controller would also need to know the data source, SAS "By" variable(s) as well as a macro variable parameter name (for each "BY" variable) that will be used to interface to the original SAS program.



And of course, some change is required to the original SAS program to properly interface with the Grid controller software.

Example

Assume that we are running a **sort** against Census data (approximately 30 million records).

Before	After
LIBNAME IN '\\Server1\Input'; PROC SORT DATA=IN.DATA1; BY PROVINCE; RUN;	LIBNAME IN '\\Server1\Input'; PROC SORT DATA=IN.DATA1; BY &Parm_Province; RUN;

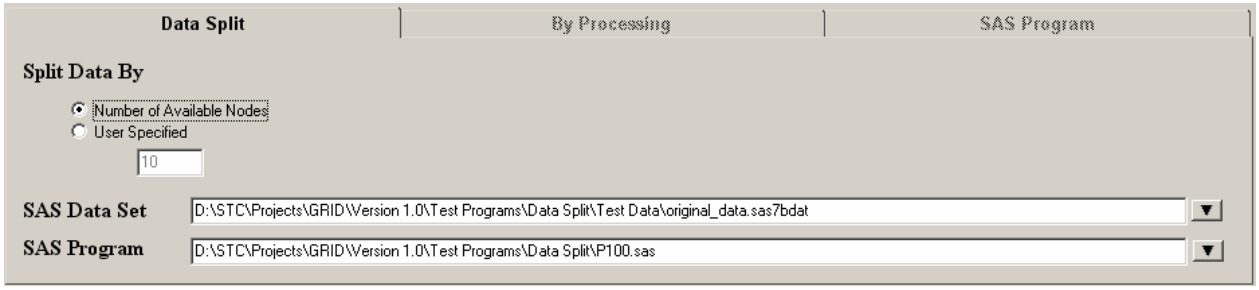
Note: Other changes may be required to the SAS code depending on how the data is to be accesses. (i.e. using SAS\SHARE®). Please see the "Other Considerations" section for more information on this topic.

In this example a SAS macro variable (&Parm_Province) replaces the original variable (PROVINCE). At run time the Grid controller software will dynamically determine the unique values of the SAS "BY" variable and create separate SAS programs. Each unique value will get passed into the program through the specified SAS macro variable.

DATA SPLIT SAS PROGRAMS

And finally, we decided to implement a third option in the Grid prototype, which we refer to as "Data Split". Sometimes data is not processed using a "BY" variable but rather at the record level. When processing data at the record level, it is possible to arbitrarily divide the data into smaller chunks of data. These smaller chunks can then be sent to the connected nodes for processing in parallel. Again, some minor change will be required to the original SAS program.

The Grid controller software will need to know the data source, the SAS program that processes the data and how to split the data. It could be a user specified value or it could be based on the number of available nodes on the Grid.



Because data will be divided and sent to nodes, a change to where the SAS program retrieves the data will be required. In the prototype, it was decided to "Upload" the data from the original location to SAS WORK on the given node. Therefore the SAS program must process the data in SAS WORK.

Example
Assume that we are processing data at the **record level** for Canada tax data (over 20 million records).

Before	After
LIBNAME IN '\\Server1\Input'; DATA **IN.TAXDATA**; TAX_DUE = INCOME - DEDUCTIONS; RUN;	DATA **WORK.TAXDATA**; TAX_DUE = INCOME - DEDUCTIONS; RUN;

In this example the SAS Library Reference needed to be changed to WORK. The Grid controller software would take care of transferring (Uploading) the data to SAS WORK on the node as well as transferring the data back (Downloading) to the original location. SAS\CONNECT Remote Library Services or PROC UPLOAD and/or PROC DOWNLOAD could be used for data transfer. The same SAS program could then also be sent to each node to correctly process the data in SAS WORK.

OTHER CONSIDERATIONS

There are a few considerations that should be briefly mentioned or addressed when dealing with a Grid Computing environment using SAS.

FOLDER DEFINITION

Due to the fact the nodes on the Grid are typically distributed across the network, referring to folder location becomes very important. On one machine (node) it may be valid to refer to a folder by a defined drive letter, however, on a different machine (node), a different drive letter may be used to point to the same folder. And of course, in a Grid environment, it is not known in advance which machine will process which data. The Grid controller dynamically takes care of this. Therefore, it is important to use UNC (Universal Naming Convention) paths when defining folder locations (in a Windows Environment).

Use	Do Not Use
LIBNAME IN '\\Server1\Input';	LIBNAME IN 'S:\Input';

ERROR HANDLING

The Grid controller software should include some sort of error handling. There are two possibilities that could happen. First, a given node could freeze, in which the client would never know if the job is still running or has froze. A second possibility is that the job sent to the node could crash (SAS errors). In the prototype, focus has been placed on handling the second type of error; errors generated by the SAS system. Before the actual SAS program is sent to a node by the Grid controller software, additional SAS code is added before and after.

```
%LET SYSCC=0;
...
... SAS Program
...
%SYSRPUT SASRC_SERVER1=&SYSCC;
```

The SYSCC automatic SAS macro variable is set to "0" at the beginning of the SAS program. If any error occurs during any of the steps or procedures, the SAS system will set the macro variable SYSCC to a value greater than 4 (=4 means SAS Warnings were encountered). And of course the value of SYSCC must be transferred back to the client machine where the Grid controller software will handle any errors accordingly.

ACCESSING DATA

The accessing of data becomes a big issue when talking about Grid Computing. Because nodes are usually distributed across a network, and the fact that several nodes are usually running asynchronously, there is a potential concurrency problem of accessing data from a central data source. A solution to this problem could be to use the SAS\SHARE software. Of course this may mean additional changes to SAS programs, in the way in which data is referenced (i.e Libnames, Filenames, etc.).

Another approach is to have the Grid controller software actually move the necessary data to each node, and re-assemble once completed. In some instances, this method may work well (Example, “Data Split” model discussed earlier).

CODE EXAMPLE

Before SAS code is submitted to a node on the Grid, additional SAS statement need to be added before and after the actual program. The following illustrates an example of some code that the Grid controller would generate and submit:

```
RSUBMIT Server1 WAIT=NO CSYSRPUTSYNC=YES;
   %LET SYSCC=0;
   ...
   ...   SAS Program
   ...
   %SYSRPUT STATUS_SERVER1=COMPLETE_1234;
   %SYSRPUT SASRC_SERVER1=&SYSCC;
ENDRSUBMIT;
```

SUMMARY

The intention of this paper is not to provide a specific recipe for a Grid Computing environment in SAS, but rather a guide that addresses many of the important issues of the concept.

Using SAS\CONNECT one can easily set-up the necessary Grid infrastructure. Using SAS\AF, SAS macros or other SAS software, the necessary Grid controller software can be developed. And finally, minor changes need to be incorporated into the actual SAS program(s) depending on how the Grid controller software was implemented.

“If it can be submitted in the SAS Editor, it is a potential candidate for Grid Computing”

CONTACT INFORMATION

For information on topics covered in this paper including the mentioned prototype please contact:

Statistics Canada Statistique Canada

Greg McLean
Project Leader – SAS Technology Centre
System Development Division
R.H. Coats Building, 14th Floor, Section Q

Ottawa, Ontario, Canada K1A 0T6
(613) 951-2396 Fax (613) 951-0607
greg.mclean@statcan.ca

Canada