Paper 35-26

Improving Productivity with MP CONNECT

Wen Lin, Baosteel Computer System Engineering Corporation,
Shanghai, China

ABSTRACT

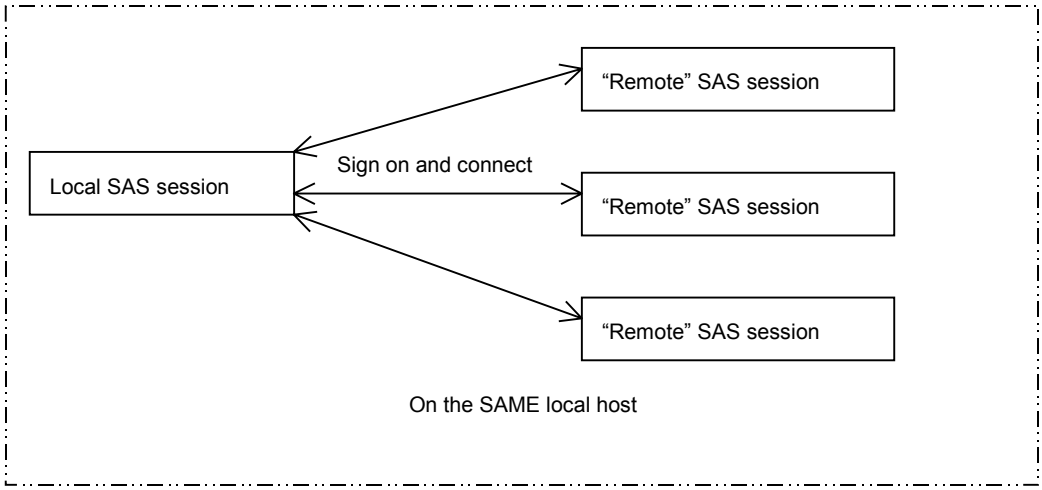
Have you ever complained that you can not have your SAS applications take advantage of the multi-processors available in your client or server platforms in order to get more done in the same amount of time? If so, and your system has already upgraded to SAS V8, then the problem can be solved easily. A new feature in V8 of SAS now allows your SAS jobs to take advantage of your MP/SMP hardware. The feature is part of the SAS/CONNECT module and is called MP CONNECT(Multi-process CONNECT). MP CONNECT allows your SAS jobs to be divided into multiple independent units of work and execute in parallel, so that these jobs can be performed in less time than they are performed sequentially.

This paper presents the concept of MP CONNECT and its benefits. It will also introduce that how the author made modification/additions to the existing time-consuming batch jobs so as to improve productivity with MP CONNECT in the daily grind. And the syntax and options to enable MP CONNECT will also be covered.

THE CONCEPT

The primary purpose of multi-processing is to perform a job in less time than it would take to execute the same job sequentially. However, since V6 SAS is a single-threaded system. One SAS session can only execute on a single processor at one time even though the other processors are in idle status at the same time.

Prior to SAS V8, SAS/CONNECT has always been a client/server tool to establish a connection from a local SAS session to a remote SAS session. MP CONNECT accomplishes multi-processing by establishing a connection to one or more “remote” SAS sessions that run on the local host. MP CONNECT’s “remote” session actually executes on the local host. This facility exploits a local host’s multi-processor capability by allowing parallel processing of self-contained tasks and the coordination of all the results into the original SAS sessions. The following graph shows the relationship between the local session and the “remote” sessions.



One task is divided into a local session and several remote sessions. All the local SAS session and the remote SAS sessions are processed in parallel on the SAME local host, but on different processors. On Windows and Unix, each SAS session has its own unique SASWORK library.

THE LIMITATIONS

MP CONNECT can extremely improve your productivity. However, not all SAS applications are appropriate to use MP CONNECT. Here are some limitations to help you determine whether you can benefit from MP CONNECT:

1. You must have MP or SMP hardware;
2. SAS V8 must be installed with SAS/CONNECT;
3. There must be no interdependencies among the units of work that are divided from one task and they can execute separately;
4. The data sources can be processed separately and independently;
5. MP CONNECT is available with the TCP/IP access method only.

THE BENEFIT

Here will give an example to illustrate the benefit the author gets from MP CONNECT. The author manages and maintains a lot of time-consuming batch jobs running at night in the daily grind. Prior to SAS V8, the author has had to let them run one by one, or invoke several SAS sessions concurrently in order to save time, even though there is a SMP hardware. With MP CONNECT, only minimal modifications need to be made to the existing jobs and the time saving is achieved. The following two tables present that two batch jobs execute in less time with MP CONNECT obviously. The first column represents the number of SAS sessions to be divided, *Total Programs* represents the total number of programs in the batch job. The *MP CONNECT* and *Serial Execution* columns show the total elapsed time for the two batch jobs to execute in mm:ss format.

These two batch jobs run on a RISC/6000 with 2 symmetrically processors with the following results:

Batch job 1:

No. of Sessions	Total Programs	MP CONNECT	Serial Execution
2	39	01:19	02:18
4	39	01:01	02:18

Batch job 2:

No. of Sessions	Total Programs	MP CONNECT	Serial Execution
2	202	35:00	45:00
3	202	30:00	45:00

The aforementioned shows a remarkable time saving using MP CONNECT instead of serial execution. With MP CONNECT, productivity is really improved.

EASY TO USE

MP CONNECT is powerful and easy to use in the mean time. The following statements present how MP CONNECT invokes a remote SAS session and a local SAS session, the operating system of the local host is AIX 4.3:

```
Options AUTOSIGNON=YES;
Rsubmit PROCESS=task1 WAIT=NO MACVAR=tt
SASCMD='/sas/sasv8/sas';
 ⋮
 ⋮   /*The programs execute in this remote SAS session*/
 ⋮
endrsubmit;

WAITFOR task1;
/*The programs execute in the local SAS session*/

DM 'RGET';
```

1. AUTOSIGNON: if the option is specified to YES, then the local SAS session will automatically sign on the remote SAS session before the rsubmit statements execute. Once the SIGNON is executed and the connection established, the programs in the rsubmit statement will perform concurrently with the other remote and local sessions;

2. PROCESS=: to specify the alias of the remote SAS session, you can also specify a session alias directly, for example, rsubmit *task1*;

3. WAIT=: rsubmit statements are processed in either synchronous or asynchronous mode.
   Synchronous mode: to specify WAIT=YES|Y, it means that the local SAS session and the other remote sessions can not execute until this remote session has completed. Synchronous mode is the default processing mode.

   Asynchronous mode: to specify WAIT=NO|N, it means that the user can start this remote session in the background and to immediately continue with local session or other remote sessions.
   In this mode, since remote SAS session executes in the background, we can not know when it will complete. There is an option which enables us to adjust:
   MACVAR=

0	Remote session is complete
1	Remote session failed to execute
2	Remote session is still in process

   In this example, *tt* is the name of a macro variable. We can know whether the remote SAS session has completed or is still in process through its value.

4. SASCMD=: to specify the command to be used to invoke the remote SAS session for MP CONNECT;

5. WAITFOR: the statement is used to make the local SAS session unable to execute until the completion of the remote SAS session. You can specify different values under certain circumstances. WAITFOR _ANY_ is to suspend the local SAS session until the completion of all the specified tasks(a logical OR). WAITFOR _ALL_ is to suspend the local SAS session until the completion of all the specified tasks(a logical AND).

6. DM 'RGET': when a rsubmit statement is processed in asynchronous mode, the log information and the output statements do not appear in the local log and output windows. The RGET command and RGET statement cause all the spooled log and output from the execution of an asynchronous rsubmit to appear in the local log and output windows. Besides DM 'RGET' statement, you can type 'RGET' command in the command line after submitting all the SAS sessions. But maybe the asynchronous rsubmit is still in process when the RGET command or RGET statement is executed. That will cause the rsubmit statement continues processing synchronously. So you can use these two statements to make sure the completion of all remote SAS sessions before you get their log and output lines:
   WAITFOR *session_name*;
   SIGNOFF *session_name*;

EXAMPLES

The following example illustrates the true beauty of MP CONNECT, how easy it is to use. The author really improves productivity of an existing batch job with this feature. The batch job includes 202 programs and it costs about 45 minutes when executing sequentially. With MP CONNECT, the batch job is divided into three units of independent work, and therefore three SAS sessions are created. Currently the batch job saves about one third time. The time saving is obvious.

```
Options AUTOSIGNON=YES;
Rsubmit PROCESS=report1 WAIT=NO
SASCMD='/sas/sasv8/sas';
%inc '...../x1.sas';
%inc '...../x2.sas';
%inc '...../x3.sas';
 ⋮
endrsubmit;

Options AUTOSIGNON=YES;
Rsubmit PROCESS=report2 WAIT=NO
SASCMD='/sas/sasv8/sas';
%inc '...../y1.sas';
%inc '...../y2.sas';
%inc '...../y3.sas';
 ⋮
endrsubmit;

%inc '...../z1.sas';
%inc '...../z2.sas';
%inc '...../z3.sas';
 ⋮
```

No special syntax is required and there is no need to do any special coding. So converting existing programs to use MP CONNECT is very easy.

CONCLUSION

If you have a lot of time-consuming batch jobs, then it is time to switch to MP CONNECT. The more processors in your hardware, the more tremendous time savings you can see. But before you consider implementing MP CONNECT, it is highly recommended that all your SAS applications be evaluated whether they are appropriate to use this feature, and whether you can get a good return on that.

REFERENCES

1. Cheryl Garner, "Multiprocessing with Version 8 of the SAS System", *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*, pp. 16-25.
2. *SAS/CONNECT Software*, *SAS Online Doc V8* .

CONTACT INFORMATION

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

Lin, Wen
Baosteel Computer System Engineering Corporation
No. 688 Fu Jin Road, Baoshan District
Shanghai, China
Zip Code: 201900
Work Phone: 86-21-26646409
E-mail: wenlin@sh163.net