Paper TR1005

Leaping Nodes in a Single Bound: Supporting SAS MP CONNECT with Windows 2000 Advanced Server

Carol A. Murphree, CISER, Cornell University, Ithaca, NY
Janet C. Heslop, Cornell University, Ithaca, NY
Simon D. Woodcock, CRADC and Department of Economics, Cornell University, Ithaca, NY

ABSTRACT

Our institute provides computing support for academic researchers, many of whom perform complex statistical analyses with extremely large data sets. We maintain a cluster of Dell PowerEdge 6350 quad processor servers running Microsoft Windows® 2000 Advanced Server. We describe the practical aspects of supporting the MP CONNECT feature of SAS/CONNECT® software so that our SAS users can take advantage of the SMP architecture of these nodes. We provide examples of user documentation and sample programs that demonstrate the use of MP CONNECT on both single and multiple nodes. We describe two research applications of MP CONNECT on our system.

INTRODUCTION

The Cornell Institute for Social and Economic Research (CISER) provides support for social science researchers at Cornell University. An important aspect of that support is to provide high performance computing resources and a range of statistical analysis software, as well as consulting and on-line help for the user community. At present we maintain 8 Dell PowerEdge 6350 quad processor servers[1]. One cluster of 6 nodes is for general access by CISER members and their research staff. The other cluster of 2 nodes is for the exclusive use of the Cornell Restricted Access Data Center (CRADC). These servers are used as "compute nodes" and each cluster is connected to a file server[2] with a minimum of 1 terabyte of disk space for user data and program storage. Software provided on each of the compute nodes includes SAS, SPSS, Stata, Gauss, GLIM, Genstat, Limdep, Mathematica and Matlab, as well as general programming and data conversion software. SAS is by far the most heavily used of all of these programs on the cluster of 6 public nodes used by our 300+ users[3]. Our aim is to provide general support for current and potential users, including sample programs designed for our system, easy reference to online help, helpdesk support (e-mail, phone, and walk-in) and hands-on workshops. Our systems administrator, systems staff, and computing consulting staff work together to provide these services. We rely a great deal on suggestions and feedback from our users for helping us to develop and improve them.

One of CISER's goals is to encourage the use of high-end applications that will allow more sophisticated users to take advantage of our computing resources, while maintaining system stability and availability for many other users with widely varying levels of computing skill and needs. The MP CONNECT feature of SAS/CONNECT, added in Version 8 of the SAS System, is one of the applications we chose to investigate for use on our system. At the time we began this investigation, MP CONNECT applications with which we were familiar were used by businesses, usually on single multi-processor nodes, and on unix operating systems. We set out to explore the usefulness of these tools for academic researchers on our Windows 2000 clusters.

In this paper we describe our initial experiences initiating and testing MP CONNECT. We've noted the minor glitches we encountered and lessons we learned in so doing. We also provide very brief summary descriptions of the first two user programs which made use of MP CONNECT capabilities on our system, and describe some of the constraints encountered and benefits gained by these early users. We have also reproduced selected portions of our current web-based user documentation, including sample programs and guidelines for use of MP CONNECT on CISER's machines. We also describe our goals for the continued development of this and more advanced support for SAS high-end applications.

INSTALLATION AND TESTING

Initiating the use of MP CONNECT was fairly straightforward (see SAS Technical Support Document TS-DOC: TS-638), and required only that we install a spawner, run as a Windows 2000 service, and edit the relevant script file on each of the eight nodes. The spawner can be started and stopped remotely but it must be installed at the console by an administrator. The upgrade from Version 8.1 to 8.2 required re-installation of the spawner at the console as well. This was notable in our case since our machines are located remotely and administered locally by our CISER system administrator[4]. A more important note is that the installer must remember to edit the sas command in the script file for the TCPIP Connection ("tcpwin.scr" located in C:\Program Files\SAS Institute\SAS\V8\CONNECT\saslink\) to reflect the appropriate command for that system. In our case this involved replacing the word "sas" with "sas.exe". This requirement is noted in the script file itself, but we mention it only as a reminder to other users.

TEST CASE ONE

[1] Each of these servers, or "compute nodes", has four 550 MHz Intel® Pentium® III Xeon microprocessors, a secondary cache of 512KB/processor, 54 GB attached disk space, 4 GB RAM, and an external bus speed of 100 MHz.

[2] Fileservers are Dell PowerEdge 2450 servers. Each fileserver has two Pentium III 733 Mhz processors with 1 GB RAM and 133 Mhz bus speed.

[3] Our general access cluster includes 6 compute ndoes and a fileserver. This cluster is available to CISER members and their research staff. A second cluster of two nodes and attached fileserver is devoted to a group of about 22 researchers who staff CISER's Cornell Restricted Access Data Center (CRADC).

[4] Our machines are located across campus in the Cornell Theory Center (CTC) and hardware support is provided by CTC staff in accordance with a Facility Maintenance Agreement. Software installation is handled jointly by CTC staff and our local system administrator.

Our initial test of MP CONNECT involved running multiple independent PROC Mixed models on a single compute node. PROC Mixed is one of the more frequently used SAS procedures by a number of social scientists on campus and, as is often the case, our user[5] needed to try a variety of models, all using the same data set, in as short a time period as possible. Her attempts to run all of the models in a series had already exceeded the capacity of our RS/6000 AIX server. Her program, as written, took more than a week to run on any one of our Dell compute nodes. At the time our cluster was not yet open to the public and there were no other users or applications with which to compete for resources. We stored her relatively small data set (15 KB) on our remote fileserver, and divided the models among four asynchronous sessions on a single compute node. Although we did not keep detailed information from the logs, we noted that each individual model took slightly longer to run in this manner than if it were run in serial mode, but the overall time savings from running all sessions simultaneously was substantial for the user. Total execution time, in this instance, was as long as the time required for running the lengthiest model independently, plus a relatively small amount of time (measured in minutes and less than an hour) that we assumed was generated by the fact that each process had to share I/O with the other sessions as they all attempted to read the same data set. We would most likely have seen a much greater time savings if we had stored the data on a local disk and/or provided duplicate copies of the data file for each process. We suspect we could have also made use of the %DISTRIBUTE macro[6] as well, since the number of models exceeded the number of processors available on the machine. The approach we used effectively captured 100% of the CPU of all four processors, thereby monopolizing the four-processor machine for a single user. We recognized the need to investigate how we would deal with this possibility when our machine was open for general access. However our goal in this case was to speed up the research process, and shortly thereafter our user successfully completed her data analysis and her PhD.

TEST CASE TWO

Our second MP CONNECT user and co-author, Simon Woodcock, had a very different set of needs and constraints for a much larger project on which he continues to work. This project makes use of a very large (> 15 million records) database on employers and employees. The goal of the project is to mask the underlying data to protect confidentiality while preserving their statistical properties. The algorithms under development for this project are complex and require significant computational resources, but lend themselves well to parallel processing. Working with a 1% sample of the data, he has tested a number of software packages, including the Cornell Multitask Toolbox^{SM} for MATLAB^{TM} (CMTM)[7]. CMTM has the appealing characteristic of providing true parallelization of tasks and does not require the user to write the program in discrete independent units, as does MP CONNECT. Unfortunately this software was unable to process even a 1% sample of the data due to its dependency on holding the entire data set in core memory during execution. SAS has the advantage of processing the data observation-by-observation, so that the entire data set is not in memory at any one time.

Woodcock has successfully written a program which makes use of MP CONNECT by dividing the sample data and distributing it among multiple sessions running independent processes. His program contains both serial and asynchronous components, which he manages by using the RSUBMIT, WAIT=, WAITFOR, and RGET options with MP CONNECT. Each remote session includes code for sophisticated data transformations that make use of the output from a number of SAS procedures including PROC REG, PROC LOGISTIC, PROC CATMOD and PROC IML. At various points in the program data are retrieved from the remote sessions, processed further in the master session, then returned to the remote sessions for additional processing. Subsets of transformed data are finally retrieved once again into the master session for post-processing and final data set concatenation. His program also involves distributing macro definitions across MP CONNECT sessions, and accessing data from the WORK library of each session by using the SLIBREF= and SERVER= libname statements options that were introduced in SAS Version 8.2. In all cases the data is stored in local disk space on the node(s) in use.

After some experimentation that involved the careful insertion of timers into his programs, Woodcock chose to use only one multi-processor node for testing his data masking algorithm. He observed that as the size of the data set increased, the time required to move data from one node to another increased more than linearly, outweighing the time savings gained by running additional sessions on remote nodes, except where processing time for these increased at even greater rates (e.g. cubic). Since the algorithms under development rarely require simultaneous execution of more than four independent processes, and since Woodcock has exclusive access to one of the CRADC nodes, there was little to be gained by using additional nodes. He also observed that the time required to read the data to and from the external file server (even though across a Gigabit switched Ethernet connection) was much greater than that required for reading the data from a local disk. This was largely due to the difference in writing to the local RAID O disk versus writing to the RAID 5 fileserver. One obstacle that had to be overcome was the tendency for the SAS workspace to fill up the local disk space, requiring the frequent use of PROC DATASETS to delete temporary data sets that were no longer needed. This was further dealt with by purchasing additional disk space for the CRADC nodes to accommodate their greater data processing needs.

STAFF SUPPORT

While Woodcock continues to test and refine his data masking program with progressively larger subsets of his data, CISER staff are making use of the information gained from his experience, along with the results of our own testing, to develop guidelines for MP CONNECT users on our system. We now have access to a real-time web based interface that gathers and reports %CPU usage and available temp disk space on each node[8]. This not only allows us to monitor the impact of MP CONNECT and other high-end applications, but it provides our users an easy way to choose nodes for remote submissions and thereby helps to level the user load across nodes. Below is a sample of our current user documentation and the guidelines we have developed to date.

USER DOCUMENTATION

Excerpts from our web-based documentation[9] for using MP CONNECT on CISER's Athena nodes are reproduced below. Some of the sample code is a modified version of code provided in SAS System Help, SAS Version 8.2.

---

[5] The authors wish to thank Michelle Montgomery Thompson, PhD, for providing the PROC Mixed models and SAS code that we modified for our initial testing of MP/CONNECT, and for suggesting that we write up our experiences to share with other SAS users.
[6] See Doninger and Tobias (2001).
[7] Cornell Multitask Toolbox^{SM} for MATLAB^{TM} is written by John A. Zollweg, Cornell Theory Center, Cornell University, Ithaca, NY. See http://www.tc.cornell.edu/Services/Software/CMTM/index.html for more information.

[8] Written by Lucia M. Walle, Cornell Theory Center, Cornell University, Ithaca, NY.
[9] The URL for our current User Documentation is http://www.ciser.cornell.edu/consulting/sasMPconnect.html

About SAS MP CONNECT:

- SAS is a single-threaded application. Generally speaking, it executes all processes sequentially on a single processor. However, a feature called MP CONNECT, introduced in SAS Version 8, greatly expanded the capabilities of SAS/CONNECT by allowing users to run multiple SAS sessions on one or more multi-processor nodes (such as the Athena nodes), all while coordinating the timing, execution, and collection of results for all sessions from one master SAS session.

- In the right cases, the use of SAS MP CONNECT can generate great time savings for SAS users. MP Connect allows the user to run multiple independent tasks in much less time than if they were run sequentially, and may also allow the user to process more data in the same amount of time.

Types of jobs best suited for MP CONNECT:

- Your project requires that you extract or process data from independent sources before combining the results.
- You are working with very large data sets that can be broken into subsets and processed separately, before re-combining for analysis.
- You're doing simulations that require many, many repetitions of one or more procedures.
- You want to run several long-running models with the same data set (Proc Mixed or Proc Catmod, for example).

Example 1: Sample Code for using MP CONNECT on a single Athena node:

```
/***************************************************/
/* MP CONNECT syntax in bold.                      */
/***************************************************/

/* first put initial code for the master  session */

options ls =75 autosignon=yes;

  libname here 'U:\Users\cam6';
  libname samples 'U:\Users\cam6\samples';

  data here.mydat1;
    set samples.data1;
  run;

  data here.mydat2;
    set samples.data2;
  run;

/* Next spawn two "remote" asynchronous SAS sessions
on the local host. We'll use these to simultaneously
sort the two data sets before merging them in the
master session. */

rsubmit sortdat1 wait=no sascmd=" 'C:\Program
Files\SAS Institute\SAS\V8\SAS.EXE' -config
'C:\Program Files\SAS Institute\SAS\V8\SASV8.CFG' ";

  options ls=75;
  libname here1 'U:\Users\cam6';

proc sort data=here1.mydat1
out=here1.sort1;
    by idno;
  run;

endrsubmit;

rsubmit sortdat2 wait=no sascmd=" 'C:\Program
Files\SAS Institute\SAS\V8\SAS.EXE' -config
'C:\Program Files\SAS
Institute\SAS\V8\SASV8.CFG' ";

  options ls=75;
  libname here2 'U:\Users\cam6';

  proc sort data=here2.mydat2
  out=here2.sort2;
    by idno;
  run;

endrsubmit;



/* get information about the remote tasks */

listtask;

/* wait until tasks above are completed */

waitfor _all_ sortdat1 sortdat2;

/* put more code for master session here */

  data here.mergit;
    merge here.sort1 here.sort2;
      by idno;
  run;
  title "Combined Data Set";

  proc contents data=here.mergit;
  run;

/* retrieve output and logs from the spawned
sessions and sign off */

rget sortdat1;  signoff sortdat1;
rget sortdat2;  signoff sortdat2;

/********************************************/
/* To terminate problem remote sessions:    */
/*   killtask _all_;                        */
/********************************************/

/********************************************/
/* FOR MORE HELP, including help using      */
/* macros with MP CONNECT, as well as other*/
/* commands and options:                    */
/* On Athena:  Go to                        */
/* Start -->Programs -->The SAS System-->   */
/* What's New in 8.2 --> SAS/CONNECT        */
/********************************************/
```

Example 2: Sample Code for using MP CONNECT on multiple Athena nodes:

Tip: Check Athena node usage prior to choosing your nodes!

```
/********************************************/
/* MP CONNECT syntax in bold.               */
/* In this example the master session is on*/
/* Athena2 and the remote sessions are on   */
/* Athena5 and Athena6.                     */
/********************************************/

options autosignon=yes;

/* put local SAS code here, and set up      */
/* process IDs for the remote sessions     */

%let host5=athena5.ciser.cornell.edu;
%let host6=athena6.ciser.cornell.edu;

Filename rlink 'C:\Program Files\SAS
Institute\SAS\V8\connect\saslink\tcpwin.scr';

signon remote=host5 script=rlink;
signon remote=host6 script=rlink;

rsubmit remote=host5 wait=no;

/* put SAS code to be run in the first remote
session here*/

endrsubmit;

rsubmit remote=host6 wait=no;

/* put SAS code to be run in the second remote
session here*/

endrsubmit;


/**************************************************/
/* optional but useful commands:                  */
/*                                                */
/*  listtask; provides information about the      */
/*     remote tasks.                              */
/*                                                */
/*  rdisplay remote=host6; creates windows for    */
/*     remote log and listing.                    */
/**************************************************/

waitfor _all_ host5 host6;

/* put more SAS code for the master session here  */

rget host5; signoff host5;
rget host6; signoff host6;


/**************************************************/
/* If you should need to terminate your remote    */
/* sessions:                                      */
/*   killtask _all_;                              */
/**************************************************/


/**************************************************/
/* FOR MORE HELP, including help using macros     */
/* with MP CONNECT, as well as other commands and */
/* options:                                       */
/* On Athena: Go to Start-->Programs-->The SAS    */
/* System-->What's New in 8.2-->SAS/CONNECT       */
/**************************************************/
```

Notes regarding speed and scalabililty of MP CONNECT on the Athena nodes:

Single node versus Multiple node applications:

- Multiple sessions run slightly faster on a single node than across nodes (since a small amount of time is required to log on to each remote node and spawn the additional SAS sessions). However, if your independent processes each take a long time to execute, the time saved for running on a single node is most likely negligible. Therefore we suggest running sessions on multiple nodes to reduce congestion on any one node.
- Time saved starts to decrease as the number of processes generated exceeds the number of available processors. If you are running more independent tasks than there are processors, you may be able to take advantage of the %DISTRIBUTE macro to allocate tasks to processors as they become available. [See References below for more information on %DISTRIBUTE]

Data location and I/O

- If you are working with a very large data set and I/O is your biggest issue, then it is generally fastest to temporarily store your data on the T:\ drive of the node on which you are working. Remember, however, that the T:\ drive is temporary disk space and is NOT backed up. Be sure to store a copy of your data in your home directory.
- Each MP Connect session has it's own WORK library. If you need to send data back and forth across sessions and/or nodes consult the SAS Documentation and References listed below for information about SAS/CONNECT Remote Library Services, which will allow you to access the work libraries of the remote sessions from your master session.
- If the time required for reading your data is not an issue, then you should read the data from your home directory on the CISER file server (usually mapped as U:\Users\<your NetID>).

To estimate the amount of time required for your job:

- In general the time required to run a job with MP Connect equals the amount of time of your lengthiest procedure, plus the time required for any pre- and post- processing that is required in the master session, plus the amount of time required for logging on and moving data across nodes.

Help and References:

SAS HELP: You can learn much more about using MP CONNECT from the references listed below, as well as from the online Help menu from within SAS.

- From within SAS: Go to `Help` --> choose the `Index` tab --> type in `connect` --> then hit `Enter`
- OR from Windows on Athena: Go to `Start` --> `Programs` --> `The SAS System` --> `What's New in 8.2` --> `SAS/CONNECT`

USEFUL REFERENCES:

- "How Fast can your SAS Programs Run? Take Advantage of Parallel Processing with Version 8 Multiprocessing Capabilities", web page from the SAS Institute
- "Multiprocessing with Version 8 of the SAS System", by Cheryl Doninger, SAS Institute, Inc. (good introduction to MP CONNECT)
- "The % Distribute System for Large-Scale Parallel Computation in the SAS System", by Cheryl Doninger and Randy Tobias, SAS Institute, Inc. (for more advanced users)

CONCLUSION

We have found that the MP CONNECT feature of SAS/CONNECT can be a valuable tool for academic researchers in a Windows Environment. We hope to continue to investigate the scalability of particular applications of MP CONNECT across nodes running Windows 2000 Advanced Server. We have observed a trend toward the acquisition of faster servers with fewer processors per node, and expect this to provide even greater incentives for users to execute parallel processes across multiple nodes. Furthermore, as the use of MP CONNECT becomes more widespread on public nodes, we anticipate increased demand for executing parallel processes across nodes, since users will not have dedicated access to all processors on a single node. We hope to work with the Cornell Theory Center to investigate the potential use of SAS MP CONNECT in batch mode on their Velocity and Velocity Plus Clusters[10]. These currently consist of 64 parallel multi-processor compute nodes each, to which our own clusters are networked, and to which our qualified users have access. We also look forward to learning to support the use of the pipeline parallel processing capabilities that we anticipate in SAS Version 9!

REFERENCES

Bentley, John E. (2001), "SAS Multi-process CONNECT: What, When, Where, How, and Why", SUGI 26 Paper 269-26.

Doninger, Cheryl (2001), "Multiprocessing with Version 8 of the SAS System", SAS Institute, Inc., http://www.sas.com/rnd/papers/connect/mpconnect0401.pdf

Doninger, Cheryl, and Palmer, Renee (2001), "Multiprocessing with Version 8 of the SAS System", PowerPoint slideshow presented by Cheryl Doninger and Renee Palmer at SUGI 26, http://www.sas.com/usergroups/sugi/sugi26/presentations/index.html

Doninger, Cheryl and Tobias, Randy (2001), "The %DISTRIBUTE System for Large-Scale Parallel Computation in the SAS System", SAS Institute, Inc., http://www.sas.com/rnd/papers/connect/distConnect0401.pdf

SAS Institute, Inc. (2000) SA*S Online Doc®, Version 8*, Cary, NC, SAS Institute, Inc.

SAS Institute, Inc. (2001), "What's New in SAS Software for Release 8.2", from *SAS System Help*, SAS Version 8.2, Cary, NC, SAS Institute, Inc.

TRADEMARK CITATION

CONTACT INFORMATION

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

Carol Murphree
Cornell Institute for Social and Economic Research
Cornell University
294 Caldwell Hall
Ithaca, NY 14853-2602
Work Phone: 607-255-1359
Fax: 607-255-9353
Email: mailto:cam6@cornell.edu

[10] Current configuration of the Velocity cluster includes 64 servers, each containing four Intel Xeon microprocessors with an internal operating frequency of 550 MHz and 2 MB L2 cache per processor. Each server has 4 GB RAM and 54 GB disk space. The Velocity II cluster includes 64 servers, each contained two Intel Pentium III Xeon microprocessors with an internal operating frequency of 733 MHz and and 256KB L2 cache per processor. Each node has 2 GB RAM and 27 GB disk space.