Use Your Cores! An Introduction to Multi-core Processing with SAS
Erik Dilts, INC Research, Raleigh, NC

ABSTRACT

Computing power has changed direction the past few years. In the past, chip manufacturers designed faster and faster processors (CPUs), but they reached the theoretical limit of that method. This is how the multiple-core CPU in today's computers came to be. A multi-core CPU is essentially the same as having multiple CPUs. SAS® 9 can take advantage of these cores in several different ways. This paper will explain how to use multi-core processing to your advantage. In addition it will demonstrate the pros and cons of the various techniques available under Windows, and when to use them.

INTRODUCTION

Multi-core processing can save a great deal of time when running large jobs or a large number of jobs. Due to the overhead involved in coordinating the flow of data between the cores, there is rarely a 1:1 ratio in the amount of time a program takes to run on multiple cores. That is, two cores will not make a job run twice as fast as one core. Some techniques for taking advantage of multiple cores require no changes at all to existing code, while others are relatively complex and require extensive modification. When trying to split one single job up among multiple cores, the timing of program submission and completion is critical. Since SAS® is a procedural language, one step must finish before the next one can proceed. When running multiple jobs simultaneously, the only timing that needs to be considered is program dependency, i.e., if program 2 depends on the completed output from program 1. If your programs have no such dependencies, then running multiple jobs at once becomes very simple.

SETUP

The simplest way to use multiple cores is with an option: THREADS. This can be done at invocation or in an options statement. Many SAS® processes will then "thread" their operations over multiple cores. Some of the procedures that can do this are SORT, MEANS, SQL, and GLM. A complete list can be found in the online SAS® Language Reference Dictionary. This method does not require any changes to existing code.

The next method of using multiple cores is multiple batch submissions. This is the standard format to batch submit two SAS® jobs:

```
START/W SAS.EXE –SYSIN "PROGRAM1.SAS"
START/W SAS.EXE –SYSIN "PROGRAM2.SAS"
```

The "/w" switch tells Windows to wait until the first step is finished before going to the next. To submit both programs at once the following format should be used:

```
START SAS.EXE –RSASUSER –SYSIN "PROGRAM1.SAS"
START SAS.EXE –RSASUSER –SYSIN "PROGRAM2.SAS"
```

Now Windows will submit both jobs immediately. The "–rsasuser" option tells SAS® to open your profile in read-only mode so as not to lock it (which generates a log warning when multiple sessions are open at once). This method also does not require changes to your existing SAS® code. The one caveat for this method is program dependency. If Program2 requires the output from Program1, then obviously they cannot be submitted at the same time. There is another consideration for this method. If you have 20 programs you want to run, and you do not use the /w switch for any of them, then all 20 will be submitted at once – and unless you have 20 cores, that won't work very well and the jobs might even run slower. This is where things get a little more complex, but one solution is to create multiple batch programs, and then use another batch program to call them one after the other.

In general you should only have one job running per core. You can find out how many cores you have via the Windows Task Manager, which is invoked using the Ctrl-Alt-Del key combination.

This is how the performance tab looks on my laptop. The split in the graph indicates that I have two cores available:
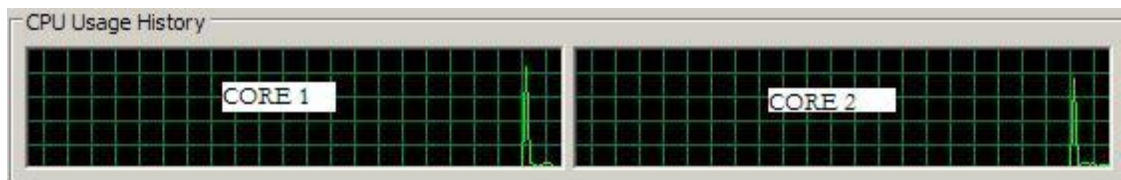

Figure 1: Windows XP Task Manager, Performance tab.

The last method I will discuss is the use of RSUBMIT within one SAS® program to submit multiple parts of it at once. This is custom programming – it is not always straightforward to modify existing code to do this. For example, you might have a very large dataset with male and female subjects, and need to analyze it with PROC MEANS. You could increase the efficiency of the program by submitting each group at the same time.

```
* create server session 1;
SIGNON TASK1 SASCMD="!SASCMD" INHERITLIB=(WORK=LOCWORK);
RSUBMIT TASK1 WAIT=NO;
  PROC MEANS DATA=DATASET;
    WHERE GENDER='M';
    OUTPUT OUT=MEANS_M;
  RUN;
ENDRSUBMIT;

* create server session 1;
SIGNON TASK2 SASCMD="!SASCMD" INHERITLIB=(WORK=LOCWORK);
RSUBMIT TASK2 WAIT=NO;
  PROC MEANS DATA=DATASET;
    WHERE GENDER='F';
    OUTPUT OUT=MEANS_F;
  RUN;
ENDRSUBMIT;

* wait for both tasks to complete before continuing;
WAITFOR _ALL_ TASK1 TASK2;

DATA MEANSOUT;
  SET MEANS_M MEANS_F;
RUN;
```

One thing to consider when using this method is that, even though it is just one SAS® session that is invoking others, they are still separate sessions, which means that parameters like options, macro variables, and library locations must be passed to each of them – they are not automatically inherited from the calling session. Timing also needs to be considered. SAS® must know that the two procedures both have to finish running before the data step runs.

CONCLUSION
SAS ® version 9 has many enhancements to take advantage of the multiple cores that exist in modern computer processors. Some of these enhancements require little to no changes to existing code, while others require that programs are written in a very specific manner. Chances are, the "extra" cores on your computer are not even being used, at least not specifically by SAS®. Multiple-core processing can greatly increase the efficiency of your SAS® jobs so you can get more work done in less time.

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

Name:	Erik Dilts
Company:	INC Research, LLC
Address:	4800 Falls of Neuse Rd., Suite 500
City state ZIP:	Raleigh, NC 27609
Work Phone:	919-334-3532
E-mail:	edilts@incresearch.com