Moving from Windows to Super-computing: The 5% Solution

AnnMaria De Mars, The Julia Group, Santa Monica, CA
Ernesto Flores, The Julia Group, Santa Monica,CA

Just think how much faster a job that takes 40 hours on a desktop could run if it used 1,000 CPUs. Many SAS users are proficient with Windows but daunted by the idea of tackling a supercomputer. This concern is largely unwarranted as >95% of SAS (r) code is identical. This paper addresses the 5% differences.

1. Changing the SAS code
2. Getting data on the server
3. Understanding home and project directories
4. Connecting to the server
5. Linux commands you need to know
6. "Sourcing" SAS
7. Problems in "tmp" space (the area where your temporary SAS datasets are stored during processing)
8. Batch and queue processing
9. Permissions - the most common problems

Yes, it really IS that easy.

INTRODUCTION

First of all, what exactly is a super-computer? There is no one set definition, in part, because capacity keeps changing and the speed that was a supercomputer thirty years ago is a lousy desktop today. The preferred term by many centers now is high performance computing. Even that is a misnomer. Our “super-computer” at the University of Southern California is actually a cluster of computers.

> Among supercomputers in an academic setting, HPCC's new supercomputer cluster is the 5th fastest in the United States and 18th fastest in the world. Among all supercomputers in the world, it is ranked 65th. It claimed this distinction by achieving a benchmark in spring 2010 of 83.29 teraflops, or 83.29 trillion floating-point calculations per second, on its 1460-node, 10-gigabit backbone cluster. (University of Southern California, 2010a).
>
> HPCC has two Linux clusters: a 896 quad-core/dual-processor node on a 10-gigabit Myrinet backbone and a 1,795 dual-processor node.of which nearly half are dual-core AMD Opteron. processors.on a 2-gigabit Myrinet network. (University of Southern California, 2010b).

Just think how much faster a job that takes 40 hours on a desktop could run if it used 1,000 processors. Processing time could be cut from days to minutes.

Many SAS® users are proficient with Windows but daunted by the idea of tackling a supercomputer. This concern is largely unwarranted as >95% of SAS (r) code is identical. This paper addresses the 5% differences.

Before you get started, there are a few things you’ll need:

1. An account on the high-performance computer. Contact the relevant department at your institution to get that.
2. A program to connect using a Linux shell. We use XWin32.
3. A program to upload and download your files. We use Filezilla.

Step 1: Changing your SAS code.

In brief, anything that is reading from or writing to a file outside of SAS needs to be changed, but that is very few statements. The LIBNAME statement using Windows references a drive name and a folder. In this example, libref refers to the folder pumpkin on your c drive.

```
libname libref  "c:\pumpkin" ;
```

On Linux you may change this one of two ways. Option one is shown below.

```
libname libref "~/pumpkin" ;
```

This will assign libref to be the directory named pumpkin within your personal home directory. Think of your home directory as your Documents folder. It really has another, longer name, like "c:\Users\annmaria\Documents" but just "Documents" will work.

On many systems you are allocated a relatively small amount of space for your personal home directory, but you may be part of a group that is allocated a much larger space. For example, if you were working with the data on Medicaid claims for the last five years, it would be an incredible waste of space for all of the ten people on this project to each have a copy of this enormous database. That's the sort of thing that would be in a project directory. To reference a project directory, you'll need a complete path, which will look something like this:

```
Libname libref "/rcf/proj/Medicaid/" ;
```

VERY IMPORTANT NOTE: While SAS is not case-sensitive, Linux is. What this means is that whether you type LIBNAME or libname is not going to make any difference but if within the quotes you type "Medicaid" and the directory is really named medicaid, all lower case, it will not be found and you will receive an error.

Anywhere you reference a file you are reading in needs to changed. This could be the INFILE statement, that needs to be changed from

```
Infile "c:\mysasfiles\fileIwant.txt" ;
```

To

```
Infile "~/myfileslinux/fileIwant.txt";
```

The FILENAME statement, if you used one, needs to be changed from

```
Filename fileref  "c:\mysasfiles\fileIwant.txt" ;
```

To

```
Filename fileref  "~/myfileslinux/fileIwant.txt" ;
```

Personally, I usually use FILENAME statements and put those at the beginning of the program. I switch between systems often and this structure makes it easy for me to change everything in one spot and the rest of my program runs …

EXCEPT ….

There's always an exception, isn't there? If, as I often do, you have several procedures throughout your program and you want to output them to various ODS destinations then you'll need to make changes throughout your program. Again, the changes are simple and painless. Change

```
Ods pdf file = "c:\myoutfiles\output.pdf" ;
```

To

```
Ods pdf file = "~/myoutfiles/output.pdf" ;
```

You can output .doc, .pdf and .html files even if you have nothing on your Linux machine that can read or open those types of files. For example, the USC high performance computing cluster was not meant to be used for web browsing and there are no browsers installed. That means that if you run your SAS job interactively (which most installations would probably frown on anyway) you won't be able to see your html file, but it is created.

What about Excel, SPSS and other types of files?

You might wonder if you could import or export file types that don't even have a Linux version, such as Excel. The answer is yes. Again, you don't need to have Excel, SPSS, JMP etc. installed to use PROC IMPORT or PROC EXPORT for those file types. You just need to change the way you reference those files. For example:

```
Proc import out=libref.test
             Datafile = "c:\myxlfiles\something.xls"
      DBMS = XLS REPLACE ;
      Sheet = sheet1 ;
      Getnames = yes ;
```

Changes to

```
Proc import out=libref.test
    Datafile = "~/myxlfiles/something.xls"
      DBMS = XLS REPLACE ;
      Sheet = sheet1 ;
      Getnames = yes ;
```

ALSO IMPORTANT TO NOTE: Just like with Windows, the directory must already exist. In each example, I'm using the same directory name as the folder on my Windows machine, but that must already be created in your Linux account before you run your program.

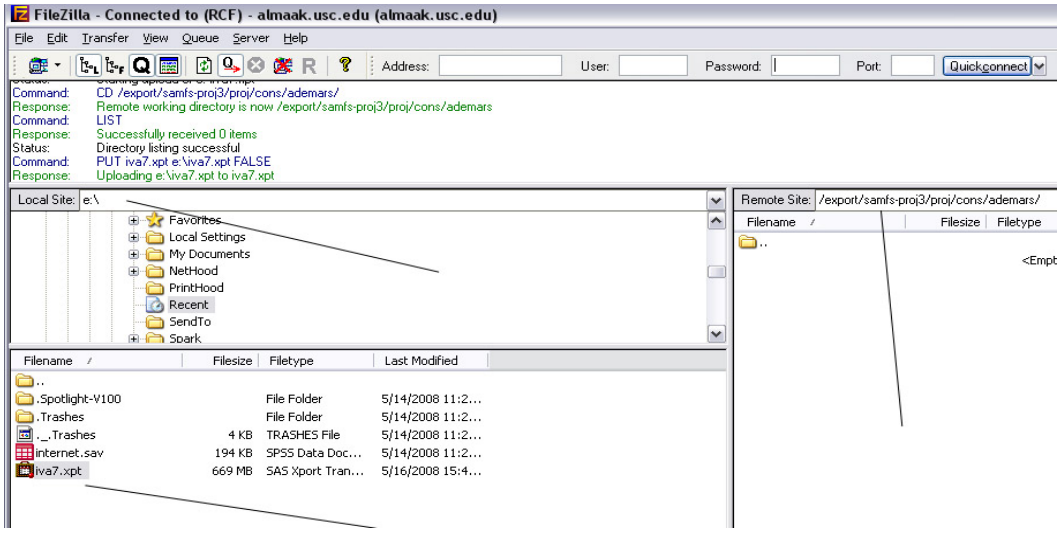
Step 2: Getting Your Data on the Server

Anything you can use to sftp (SSH file transfer protocol) will work. The example below uses Filezilla, which we have found to be the simplest. For address, enter the name of your computer. You'll need to get this from your organization. It will be something like hpcc.myu.edu . User is your userid. For port, enter 22, this is the standard port for SFTP. Click Quickconnect.

In the left pane, you'll see your local computer. Under local site, select or type in where your data are located that you want to upload. In my case, these are on my travel drive, the e: drive. Datasets in that folder will show up below

IMPORTANT: on the remote site select where your data should go, the default will be your home directory. This may or may not be where you want your data uploaded. Second important point, each time you use Filezilla, this default re-sets to your home directory. So if you had another directory when you used this yesterday and you go back today it will once again have your home directory as the remote site.

Once you have the Local (from) and Remote (to) sites set up, then just double click on the file that you want to move.



Step 3: Understanding home and project directories

Your home directory is your personal space. Think of it like the hard drive on your desktop computer. It has your stuff. Normally you don't need to do anything special to write to it, open files or delete files and no one else has access to it. The two problems with your home directory are that it usually isn't very big and it isn't a good choice for sharing files with others.

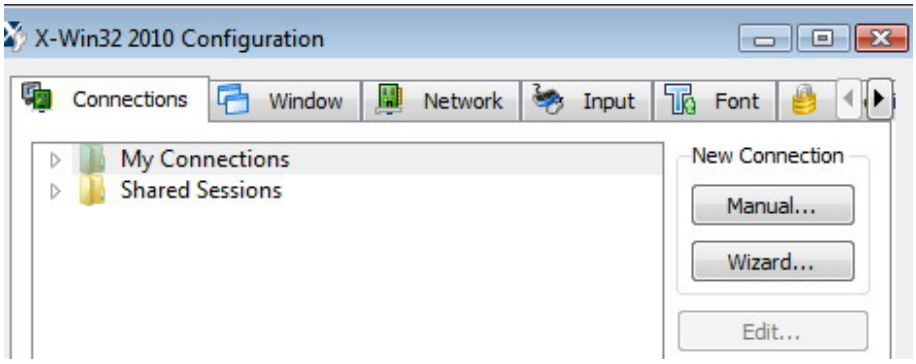
A project directory is not "yours" even though it may even have a subdirectory with your name on it. I'm going to belabor this point because I have seen it drive people crazy. How can you not have access to your files that are in your subdirectory of the project directory? Simple, really. If I created a folder on my laptop with your name on it, you wouldn't automatically be able to see it would you? Think of this like a network drive on a windows server. It has a lot more space, which is nice, but someone has to give you access to it. Often project directories have multiple users who can read, write and execute files. Be aware of this before uploading sensitive or potentially embarrassing information.

Step 4: Connecting to Your Server

Great, you have your SAS program modified, your program and your data on the server, now what? You need a program that enables you to connect to your server. On Windows, we use X-Win32 simply because it is very easy to use and you can download a free 30-day trial if your institution does not have a site license.

1. Download and install X-Win32
2. From the START menu, select ALL PROGRAMS then INTERNET TOOLS then X-WIN32 and, finally, under that, select X-CONFIG

The window below will pop-up



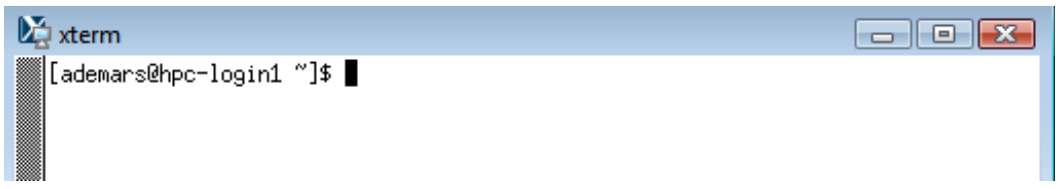
3. Select WIZARD and click OK
4. Give your connection a name, like hpcc
5. There will be four types of connections to choose from. Click on SSH and click NEXT
6. You will be asked for a host name to connect to. This is the name of the high performance computing installation at your organization. It will be something like hpc.myu.edu . Fill in the host name and click NEXT.
7. You will be asked for your user name and password. Enter that information.
8. The final window asks you for a command. This is the point where you wonder what the heck you are supposed to put. It also has a window underneath where you can click on a number of choices such as Linux, Solaris, etc. Select Linux. For command, type:

```
/usr/bin/xterm -ls
```

(This command will start a terminal emulator. The –ls specifies that the shell started in the xterm window will be a login shell.)

9. Click FINISH
10. The new connection name should show up in your window under Shared Sessions. Click OK.

You only need to do the steps above once. Now you have configured your session, in the future you can just X-Win32 from the START menu. When you double-click on it, the window will appear showing all of your sessions. Click on the hpcc session and click the LAUNCH button. A window appears that looks like the one below. You are now connected. You can now type in Linux commands – if you know any.



Step 5: A Little Linux

It's probably a good idea to get at least a limited familiarity with Linux, but there are seven commands you are going to run into sooner rather than later:

Task	Linux Command	Windows Equivalent
Log in:	ssh –Y username@hostcomputer	Enter name & password on user login screen
Create a directory:	mkdir directory-name	New folder
Change directory:	cd directory/subdirectory	Double-click on a folder to open it
Edit a file:	pico filename.type	Double-click to open or right-click and choose program
List files and directories :	ls ls – a	List of files & folders in a folder - include hidden files
Invoke sas:	sas	Click on SAS in the start menu or run from the command line
End a session	Logout	Shut down from start menu

Most Linux commands have options in the form

```
command  -option
```

One particularly important option of the ssh command is –Y. When you log in, if you just type

ssh annmaria@hpcc.myu.edu

you'll be logged in all right. However, that –Y option is what allows you to use the windowing environment of SAS on Linux. If you omit it, you can log in, use other Linux commands to create directories, list files and so on. You can even submit a SAS job in batch. BUT …. If you type

```
sas
```

to start an interactive session, instead of seeing the familiar SAS windows pop up, you'll be treated to several lines of ugly error messages, like this:

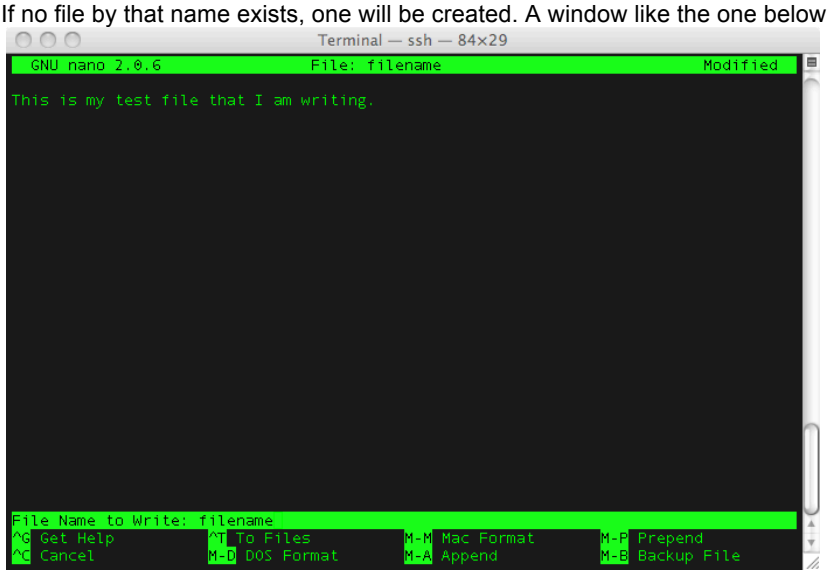
ERROR: The connection to the X display server could not be made. Verify that
ERROR: the X display name is correct, and that you have access authorization

If that happens, logout and log back in using ssh –Y. Problem solved.

In the next step, I'm going to tell you to edit a file. How do you edit a file? There's no Word on Linux. There are several options for file editing. Ernie prefers pico. AnnMaria prefers emacs, but since Ernie wrote this section ….. type

```
pico filename
```

If no file by that name exists, one will be created. A window like the one below will appear and you can just type away.



To exit from Pico type: CTRL + X
You will receive a prompt if you want to save the file. Type “Y” or “N” and click ENTER to save.

Points to remember on Linux commands

LINUX IS CASE-SENSITIVE !!! We mentioned this in passing before but we are mentioning it again because it tends to drive SAS users crazy as we are not used to it.

There is no “help menu” in Linux in a high performance computing environment. There are the “man pages”. Thankfully, women can use these as well. Man is short for manual. To get help on any command, simply type ‘man’ and the command name, e.g.,

```
man ls
```

In our opinion, the man pages were written as if the authors were charged by the word for writing them. They are definitely not overly loquacious. They are, however, always available.

Step 6. "Sourcing" SAS

You logged in. You created a directory for your SAS files. You typed SAS. Nothing happened.
Now what?

Remember we said that Linux is case-sensitive? Try typing sas
Still nothing?

You may need to provide the location where SAS is installed in your .login file. This is going to be specific to your organization, so you’ll need to email or call the department responsible for your high performance computing center and explain that you need to be able to add a source statement for SAS to your .login file. [See how you are already sounding like you are on top of things?]

If SAS is not working and there is not already a source statement that references SAS in your .login file, do this:

Log in to your account.
Type

```
pico .login
```

As the last line in the file, add the statement that your organization provided. It should be something like this:

```
source /usr/myu/sas/9.2/setup.csh
```

Log out.
Log back in.
Type

```
sas
```

It should start a SAS session with the log window, program editor window, output window and you’re back in familiar territory.

To submit a SAS job to run in batch, just type

```
sas filename.sas
```

Step 7: Problems in "tmp" space

The tmp directory is the area where your temporary SAS datasets are usually stored during processing. Just like you may have two types of directories – your own personal home directory and a shared project directory, there are often two types of tmp directories, your own personal tmp directory and a shared one.

Depending on how your particular system is set up, it is possible that you can skip this step. IF the default for the working directory is the shared tmp space and IF there is a large amount of space available OR IF you are not working with very large datasets, then you’ll be fine. What if one of those conditions is not met?

Generally, your personal tmp space is going to be small and easily taken up. If you run a job with an enormous dataset, it will fail. Even if you are using shared space, because it is shared and it is possible that someone might store a huge file there just temporarily and hog up most of that space (okay, that might have been me, sorry).

To run SAS and specify a directory other than the default, use the –work option on the SAS command. For example:

```
sas myfile.sas –work /dir1/dir2/
```

will run the program myfile.sas using as your working directory the directory dir2, which is a directory of dir1 . Note that this directory must already exist and you must have write access to it. (More about that below).

Step 8. Batch and queue processing

Most SAS users are accustomed to thinking of, at most, two means of submitting their jobs. The simplest is interactive processing. You have your SAS program open, you click on the little running man icon in the top window and your program runs.

A second method is batch. In batch mode, no windows are opened. Your SAS statements, let's assume they are in a file named example.sas are executed. The log statements are saved in a file named example.log . What would, under an interactive mode, appear in your output window, is saved in a file named example.lst .

If you simply type 'sas' and the SAS program file name, your job will run in the foreground. That means that you cannot execute any other commands until the SAS job has completed. If it is a job that runs really quickly, you may not even notice, as the delay may just be a second or two. However, for longer jobs, or those of you who are really impatient, you can also run the job in the background by adding an ampersand, e.g,

```
sas myfile.sas &
```

Now you can type other commands and continue merrily along with your work while SAS runs in the background.

High performance computing can be a fabulous resource and just like other fabulous resources, you can end up with multiple people who all want to use it at the same time. Therefore, most installations will use some type of job scheduling system. A common one used for Unix clusters is the Portable Batch System (PBS).

To submit a job using PBS, you need to create a PBS file. Fortunately, this is really, really easy. Just use pico (aren't you glad you learned pico now?) and create a file named whatever.pbs

All that needs to be in this PBS file to run your job is a single statement:

```
sas /dir1/dir2/filename.sas
```

That's all that NEEDS to be in there but it is likely you will want more. You may wish to allocate more memory to your job. The default memory allocation for Unix is 128M. If you have a very large job you may get an out of memory error. No worries, just increase the memory size.

```
sas –memsize 512M –work /mydir/project/sasdir/  /mydir/project/sasdir/bigfile.sas
```

The command above allocates 512M of memory to this job, sets the working directory and then specifies the SAS file with the statements to run. To submit your job to the queue, you use the qsub command

```
qsub whatever.pbs
```

Your system will give you a response something like this, which lists the job identifier

876543.myu.hpc.edu

To see the status of your job, use the qstat command followed by the numeric job identifier, e.g,

```
qstat 876543
```

You should see something like this:

```
Job Identifier      Name          User   Time Use      S      Queue
876543.myu    whatever.pbs  ademars              0      Q      quick
```

This job has not run yet and is assigned to the "quick" queue, which is for short jobs that will require less than 30 minutes of wall time (that is time as measured by the clock on the wall, not CPU time). When you have more experience with Linux and high performance computing you'll no doubt move to larger jobs and require additional resource in time, multiple nodes. When you get to that point, the USC High Performance Computing and Communications (2004) website offers a user-friendly tutorial on PBS that should answer most of your questions.

Step 9: LINUX Permissions

Now you've learned how to modify your SAS code, upload your data to the server, how to use a few Linux commands, some basic Linux concepts and how to submit your job using the scheduling software. We've even tried to anticipate and prevent some common problems with limits on memory size and temporary space. Congratulations!

Still, Murphy's Law tells us that there will be problems, and our experience tells us that the most common problems involve permissions.

How to check permissions on files and folders in LINUX.

You can use the ls command with the -l option to show the file permissions set.
For example, for ernie.txt, I can do this:

```
$ ls -l ernie.txt
-rwxr--r--   1 july 01 2008 ernie.txt
```

The first space has a d if this is a directory, otherwise it has a -. The next three spots are for r(ead), (w)rite or e(x)ecute permission for you. The three after that are rwx for the group and the three after that are rwx for the whole world.

How to change permissions on files and folders in LINUX.

Each digit of this code sets permissions for one of these groups as follows.
Read is 4. Write is 2. Execute is 1. The sum of these is 7, so a 7 in a given field means that segment of the population can read, write and execute. For example, 700 means that only you can read, write to, or execute ernie.txt. To change your permissions, you use the CHMOD command.

The example below changes the permissions for the file, ernie.txt and then does an ls to see the new permissions.

```
$ chmod 700 ernie.txt
$ ls -l
   1 -rwx------   1 july 01 2008 ernie.txt *
```

A very common type of permission is 755. This means that only you can write to ernie.txt. Everybody can read or execute ernie.txt

```
$ chmod 755 ernie.txt
$ ls -l ernie.txt
   1 -rwxr-xr-x   1 july 01 2008 ernie.txt *
```

Other options

chmod 444 ernie.txt	Everybody can read ernie.txt.
chmod 777 ernie.txt	Everybody can read, write to, or execute ernie.txt
chmod 744 ernie.txt	Only you can read, write to, or execute ernie.txt Everybody can read ernie.txt;

While a complete discussion of permissions is beyond the scope of this paper, suffice it to say that this is a topic worth learning that will prove surprisingly useful.

Conclusion

This very brief, rapid-fire introduction has highlighted the main points you absolutely need to know to move from Windows to super-computing. Now for our truth-in-advertising disclaimer: Every person we have worked with on moving their processing to the high performance computing cluster has spent from a couple of days to two weeks, calling us, swearing, becoming frustrated and probably becoming frustrated and swearing at us (usually not to our faces). You are guaranteed to forget several times that Linux is case-sensitive and have commands not work or do unexpected things. There will inevitably be Linux commands you need that we did not covered here , like the kill command (and no, that does not take out your enemies). We did not explain how to delete those files you created, how to move or copy datasets. You have the man pages for that, or, two books we have found more useful, Teach Yourself Unix in Ten Minutes and Unix for the Impatient.

All of that being said, once those few days (or weeks) are over, you will never believe how you lived without this kind of computing power. Jobs that took hours to run will now be completed in minutes. Computing at a level you could not have previously imagined – analyzing millions of web pages, hundreds of millions of posts or tweets – can now be accomplished using the same SAS statements with which you are already completely familiar. The parallel processing is done automatically by SAS. You don't need to change a thing in your programs other than the few statements we reviewed above.

There! You have no excuse. So, move on to super-computing and say hi to HAL for us.

References

Abrahams, P. W. & Larson, B.A. (2006. Unix for the Impatient (2nd edition). Addison-Wesley Professional.

Shimonski, R. (2005). Sam's Teach Yourself Unix in Ten Minutes. Sam's Publishing

University of Southern California (2004). Portable Batch System. http://www.usc.edu/hpcc/systems/use-l-4.php Retrived from the Internet September 21, 2010.

University of Southern California (2010). High performance computing and communications. Retrieved from the Internet August 21, 2010 http://www.usc.edu/hpcc/

CONTACT INFORMATION

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

Name: AnnMaria De Mars / Ernie Flores
Enterprise: The Julia Group
Address: 2111 7th St #8
City, State ZIP: Santa Monica, CA 90405
Work Phone: (310) 717-9089
E-mail: annmaria@thejuliagroup.com
Web: http://www.thejuliagroup.com