A Beginner's Guide to HPC with R on Nautilus

National Institute for Computational Sciences

July 3, 2013

Contents

1 Introduction

Herein, we hope to ease the transition for R users moving from the desktop to the Nautilus supercomputer. Knowledge of R as well as some basic principles of parallelism are assumed; however, no particular mastery of various parallel R API's is assumed.

This document is ideally suited for the user who has only or mostly done R development on a single, standalone workstation (laptop/desktop computer). This is by no means meant to be complete or comprehensive, and is primarily focused on bridging the initial gap of knowledge in moving from a workstation to a remote system for the R user.

1.1 What is Nautilus?

Nautilus, an SGI Altix UV 1000 system, is the centerpiece of NICS Remote Data Analysis and Visualization Center (RDAV). It has 1024 cores (Intel Nehalem EX processors), 4 terabytes of global shared memory, and 8 GPUs in a single system image. Nautilus currently has a 427 TB Lustre file system, a CPU speed of 2.0 GHz, and a peak performance of 8.2 Teraflops.

The primary purpose of Nautilus is to enable data analysis and visualization of data from simulations, sensors, or experiments. Nautilus is intended for serial and parallel visualization and analysis applications that take advantage of large memories, multiple computing cores, and multiple graphics processors. Nautilus allows for both utilization of a large number of processors for distributed processing and the execution of legacy serial analysis algorithms for very large data processing by large numbers of users simultaneously.

1.2 Getting an Account

See the allocations and accounts page on the NICS site for details.

2 Interacting with Nautilus

For those not yet familiar with computing on remote systems, the first big hurdle to joining the HPC world is to learn the basics of connecting to and interacting with a remote system. In the section to follow, we hope to dispel some of the mysticism for interacting with Nautilus from your workstation.

A complete set of documentation on connecting to Nautilus can be found at the the access page on the NICS site. We will provide some examples that should illustrate the process sufficiently for users, but if there is ambiguity, you are always encouraged to check the NICS site documentation.

2.1 Connecting to Nautilus

In order to connect to and use Nautilus, you must have a **s**ecure **sh**ell (ssh) client. SSH is a protocol that allows encrypted connections between remote computers. It consists of two parts, the ssh server (Nautilus) and the ssh client (your computer).

2.1.1 Connecting from Mac and Linux

Mac OS X and Linux have a ssh client bundled with the operating system. On a Mac, you can open a terminal by navigating to your Applications folder, then the Utilities folder, and running the app "Terminal". On Linux, the method for starting a terminal will depend to some degree on your distribution of choice.

If you would prefer, you can also use Putty (see section 2.1.2 below) as it is multiplatform (although this is not necessary). It provides a graphical user interface for starting (but not using) a terminal session. For the remainder, we will not assume that you are using Putty (though if you are, it should be reasonably clear how to take the information here and use it appropriately in Putty; alternatively, see section 2.1.2 below).

For demonstration purposes, say your username for your Nautilus account is `nautuser`. Exactly how you connect will depend on whether you have an XSEDE account or a Director's Discretion account from section 1.2. If you have an XSEDE account, then you would enter into the terminal:

```
ssh nautuser@login.nautilus.nics.xsede.org
```

On the other hand, if you have a Director's Discretion account, you would login by entering into the terminal:

```
ssh nautuser@login.nautilus.nics.tennessee.edu
```

Of course, it is inconvenient to type this into the terminal every time, so you may wish to configure a "shortcut" of sorts. To do this, you would need to create a file called `config` and put that in the `.ssh/` (note the preceding dot) subdirectory of your home directory. If this directory does not exist, create it. With your text editor of choice, you might enter into this config file:

```
Host nautilus
HostName login.nautilus.nics.xsede.org
User nautuser
TCPKeepAlive yes
```

filling in the appropriate information as needed, so that to connect you need only type

```
ssh nautilus
```

into the terminal.

However you choose to go about it, begin an ssh session with Nautilus. On your first login, you will be prompted about adding a RSA key to your cache. Do so.
See section 2.1.3 below for complete details about your password.

2.1.2 Connecting from Windows

Windows users will have to download an ssh client, such as Putty. For a complete set of Putty documentation, see the putty user manual.

After starting putty, you will need to enter the Nautilus host name under the Session category. Make sure that the connection type is set to SSH and that the port is 22. Which host name you use will depend on which type of account you created in section 1.2. If you have an XSEDE account, then you would use the hostname

```
login.nautilus.nics.xsede.org
```

On the other hand, if you have a Director's Discretion account, you would login by entering into the terminal:

```
login.nautilus.nics.tennessee.edu
```

You can save this session either as the default or under the name of your choosing. There are many configurable options available to you in Putty, but they are not needed to proceed (though altering them may make your experience more enjoyable).

Your first time connecting you will be greeted with a warning about a secure RSA key. Select the option to add the key to Putty's cache. You will then be greeted by a prompt asking what name you wish to login as. Enter the account name from the account creation step and press enter. You will then be asked to enter your password. See section 2.1.3 below for complete details about your password.

2.1.3 Your Nautilus Password

Your password for the Nautilus system will consist of two pieces: your PIN that you set in the account creation process, and the number that shows on your NICS token. For illustration, say your PIN is 1234 (do not make this your PIN) and your NICS token reads 567 890. Then the password you would enter when prompted would be 1234567890.

The number displayed on your NICS token will change roughly every 30 seconds (the little bars on the lower left will give you a sense for how much longer you have with that particular set of digits). For more information, see the access page on the NICS site.

2.2 Basic Commands

Once you are connected to Nautilus, you will be greeted by a text prompt. If this is your first interaction with such a system, you may have no idea how to proceed. This part can be confusing at first, but with a little time and patience, it will become second nature to you.

When interacting with the shell, never forget that it is case sensitive. The table below provides the basic

command	description	example
ls	list files in a directory	ls
mkdir	make directory	mkdir dir
cd	change directory	cd dir
cp	copy a file	cp a b
mv	move and/or rename a file	mv a dir
man	the help system	man mv

commands you will need most of the time.

Never be afraid to read the man pages. If you ever find yourself wondering "is it possible to", the answer is yes, and it's probably an option explained in the man page for the program. To search within a man page, use the `/` key followed by your query. If there are multiple hits for your query, you can jump to the next one by pressing `n` and to the previous one with `N`.

2.3 Loading Software

Much of the software available on Nautilus must first be loaded by the user before it can be used. If you enter the command

```
R
```

into the terminal (remember, case sensitive), you will get the message

```
-bash:  R:  command not found
```

assuming your shell is bash. At any time you can see which shell you are using by entering the command

```
file /bin/sh
```

So how do we start R? You must first load R through the module system. To do so, you would enter the command

```
module load r
```

into the terminal, and R will now start as expected after entering the "R" command into the terminal. You only have to use the module load command once per login session (but you must load R again after a disconnect).

```
module load r
R
```

The module system is very powerful and very important to your life on this (or any other) supercomputer. Entering the command

```
man module
```

will give you a complete description of the functionality and use of the module system. For more information, see the Nautilus software page on the NICS site.

The table below summarizes the use of the module system.

Command	Purpose
module avail	List available programs in the module system
module load <program>	Load program
module unload <program>	Unload program

2.4 Choosing a Text Editor

Choosing a text editor can be a very lengthy journey, somewhat akin to seeking out religious enlightenment. No one text editor is perfect, and a discussion of the features of common text editors is well beyond the scope of this document. Eventually, you will likely want to choose use either Emacs (no relation to Apple) or vim. Each of these is available on Nautilus each time you log in (you do not have to load them through the module system). If you are not familiar with either of these editors, they can be very difficult to learn at first, but learning to use one of these well is a good idea.

For now, these might be a bit intimidating and act as one more hurdle in getting to the real work you want to do. To that end, it might be a good idea to start with a more friendly text editor, such as nano. Nano is not loaded by default each session, so you will have to enter the command

```
module load nano
```

to first load nano, and then call the editor by entering

```
nano
```

The commands for nano are visible at the bottom of the screen. So here, Ctrl together with R reads in a file, Ctrl together with O saves, etc. The editor is extremely minimalistic in terms of features, but is a fine place to start until you become more comfortable working in a terminal.

If you wish to have nano loaded each time you log in to Nautilus, you could start nano, enter Ctrl+R to bring up the read file dialogue. For the file, enter

```
~/.bashrc
```

and somewhere inside this configuration file, add the line

```
module load nano
```

Press Ctrl+O to save the edit, and whenever you log in from now on, you do not first have to first load nano through the module system to be able to run nano.

2.5 Transferring Files

To transfer files over to Nautilus, you have a variety of options, explained in depth at the Data Transfer page at the NICS site. For getting started, probably the two methods of file transfer available that will be of interest are sftp for small files and GridFTP for big files.

2.5.1 Small File Transfer

If the files you wish to transfer are not particularly large (eg, small datasets, some R scripts, etc.), probably the easiest way to proceed is to connect to Nautilus by sftp (the secure file transfer protocol).

Mac and Linux users can use the terminal as with ssh, and Windows users can use Putty (via PSFTP) to connect via sftp. Although there is gui program available for managing files over sftp called filezilla. If you elect to use filezilla, you should read its documentation, although the program should be fairly self-explanatory.

To connect from a terminal (Mac/Linux), you can enter the command `sftp` as you would use `ssh` in Section 2.1.1. So for example, you might enter the command

```
sftp nautuser@login.nautilus.nics.xsede.org
```

or if you set up your ssh config file as in the example, you could do

```
sftp nautilus
```

From here, you can transfer files via the self-explanatory commands `put` and `get`. The above applies for Windows users, after you replace "sftp" with "psftp".

Your sftp password is the same as your ssh password.

2.5.2 Large File Transfer

If you need to transfer large files, such as your full dataset, using sftp is not recommended. In this case, the preferred method is GridFTP. You will need a special GridFTP client such as globus-url-copyh or uberftp. For details about how to set up and use GridFTP with these clients, see the GridFTP page on the NICS site.

3 The Job Queue System

Generally, you will do very little direct interaction with Nautilus. You should not really be using Nautilus in the same way that you use a workstation; it is not a workstation. Nautilus is a shared system. No one person is the sole user of Nautilus at any given time (usually).

The way you will want to run your analyses is by submitting your job to a queue system on Nautilus. This system handles all of the user requests and keeps the various demands for resources (cores, ram) from different jobs from stepping on each others toes, so to speak.

A very thorough explanation of the various options and ways of interacting with the job queue system is provided at the batch scripts page on the NICS site, and so we will not reproduce that information here. We merely provide a quick sketch of the general process. For full details and explanations, see the NICS site. However, you can find some example scripts in the Examples section, Section **??**.

However, there are a few options which you should be made aware of. First, processor allocations are given by node, not by core. On Nautilus, there are 8 cores per node, so when requesting processors for your job, you should generally request multiples of 8 (because that is what you are going to get anyway). If you request 1 core, you will get 8; requesting 9 gets you 16, etc. Likewise, since allocations are by node, ram is allocated similarly. Each node has 4gb of ram per cpu, and so if you request 8 cores, you will get 32gb of ram.

As for dealing with the queue system, the first step is usually to create an appropriate batch file for your particular job. This batch file will contain the information Nautilus needs to run your job. The kind of information you will need will depend on what kind of job you wish to submit. See the examples in section **??** as well as the batch scripts page on the NICS site.

Once you have your job file prepared, you can submit it to the queue using the `qsub` command. So if you have a job file called `jobfile.pbs` prepared and you are in the directory of this file, then you can submit it to the queue system via the command

```
qsub jobfile.pbs
```

You can check on your job via the commands `showq` and `qstat`, but will probably want to use the job file options to send out emails to you.

Finally, if you ever need to prematurely kill a job (whether it is running or merely queued), you can do so with the `qdel` command.

4 R on Nautilus

Performing an analysis with R on a remote system can be challenging at first. The aim of this section is to help with the transition of specifically running R for analysis on a local desktop system to running R on a remote parallel system.

4.1 Packages and Libraries

4.1.1 Using the Existing Library

Installing packages, especially complicated ones like Rmpi and gputools, can be difficult on a remote system, especially for the R user who may have never had to deal with compiling things before. Thankfully, many of the packages you will want to use are already installed for you. If we are using R version 2.12.0, then the default library path is

```
/sw/analysis/r/2.12.0/sles11.1\_intel11.1/R-2.12.0/library}
```

So merely loading R via the module system and then, for instance, if you want to use the `multicore` library, since this is already in the default library, all you have to do is

```
library(multicore)
```

If you require an additional package which is not currently installed, the simplest way to get this is to fill out a request at the Request Software Installation page at the NICS site.

4.1.2 Managing Your Own R Library

You can also of course manage your own R library, though for most this will at best be unnecessary, and at worst a needless, frustrating exercise. If you do need to manage, at least in part, your own R library, then you must install packages from source, either by downloading the source package (say with the utility wget) and install it with the command

```
R CMD INSTALL [options] packages
```

or by using the `install.packages()` command in an interactive R session, setting the various options as needed (`lib=`, `INSTALL_opts=`, . . .). One option that is not optional (without overriding an R environment variable; see paragraph to follow for details) when installing a package to a custom, user-managed library is the `lib=` argument. This is so because you do not have write access to your default library. Similarly, to load a package from a custom library, make sure you set the appropriate `lib.loc=` argument when issuing the `library()` command in R.

One additional note to consider when maintaining your own R package library is that you will likely need to manually set a few R environment variables in order to get some packages to install, or even load. For a mostly complete list of R environment variables, enter the command `help("environment variables")` in an interactive R session.

4.2 Different R Versions

For more information, see the R page at the NICS site.

4.3 Running Batch Jobs

As mentioned in Section 3, you generally should not be running R in quite the same way that you would run it on your workstation. On your workstation, you probably use R interactively; that is, you load up an R session and submit lines to the R terminal as you need.

By contrast, on Nautilus, you will be running R scripts in batch. That is, you will not start an interactive R session. Instead, you will issue a command from the terminal (really, from you job script) to start R, run your analysis, direct output to a file, and kill R when the script completes (or encounters an error). Now, for the purposes of appropriate resource allocation, this should be be done in your jobfile that you submit via qsub, rather than directly handled in the terminal. This is merely intended to illustrate what is actually going on.

One way to do this is to issue the terminal command `R CMD BATCH`. So say the script you wish to have R run is called `myscript.R`. Then issuing the command

```
R CMD BATCH myscript.R
```

will run the `myscript.R` script through R in a way that is somewhat equivalent to starting an interactive R session and issuing the R commands

```r
sink("myscript.Rout")
source("myscript.R")
q(save="no")
```