ASTR4004/ASTR8004
Astronomical Computing
Lecture 04

Christoph Federrath

9 August 2016

Remote Computing, Plotting and Movies

1 Advanced methods for remote computing

1.1 Setting up an ssh tunnel

1. If you are connected to the VPN (see notes from last lecture), you can directly connect to `misfit.anu.edu.au`. However, if you are outside the ANU network and you cannot use the VPN client for some reason, there is still another possibility to connect to the MSO servers. This requires you to first connect to a specific server at MSO that is accessible from the outside world. This server is called '`msossh1.anu.edu.au`'.

2. In order to connect to `misfit.anu.edu.au`, we simply connect to `msossh1.anu.edu.au` first and then `ssh` from `msossh1.anu.edu.au` to `misfit.anu.edu.au`. But in order to make our life easier, especially when copying files between remote computers, we can set up an `ssh` tunnel. Do this by adding/modifying the following lines in your `.ssh/config`:
```
Host misfit
Hostname misfit.anu.edu.au
ProxyCommand ssh -q -a -Y [your_mso_username]@msossh1.anu.edu.au nc %h %p
User [your_mso_username]
```
Save and close `.ssh/config`.

3. Before we can tunnel, there is one more thing that we need to do, because the program `nc` (netcat) called in the `ProxyCommand` is in a somewhat unusual place on `msossh1`. So, please login to `msossh1.anu.edu.au` and add this line to your `.bashrc`:
```
export PATH=$PATH:$HOME/bin:/opt/csw/bin:
```
This will add the directory `/opt/csw/bin/` to your `$PATH` environment variable where `nc` is installed.

4. Now you should be able to connect to `misfit.anu.edu.au` by using `> ssh -Y misfit` directly from your local computer. It will probably ask for your password twice; the first time when it connects to `msossh1.anu.edu.au` and the second time when it connects from there to `misfit.anu.edu.au`. So what this effectively does is that we tunnel through `msossh1` to `misfit`.

5. As earlier, add a 'misfit' alias to your `.alias` file as a shortcut for `ssh -Y misfit`.

1.2 Copying files/data across remote computers

1. With the changes to `.ssh/config` that we made earlier, it is now also very simple to copy files between your computer and the remote host. Lets copy the density PDF data file from assignment 1 into your home directory on misfit. First, download the data file to your local computer (if you don't have it yet) from http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/material/mM4_10048_pdfs/EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data

2. Now copy the data file to `misfit` using `scp` ('secure copy'):
   `> scp EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data misfit:`
   Note that you can use the tab key to auto-fill the rest of the awkward filename. If you just provided the initial few letters 'EX' and hit the tab key, the shell should automatically complete the rest of the filename. Note also that when you start `> scp` and then hit the tab key twice, you will be given a list of possible options for files in this directory. This is a very useful auto-completion function of the shell. Once you executed the `scp` command, you should see the file being transferred to misfit. Note that the colon behind 'misfit' refers to your home directory on misfit. If you want to copy the file somewhere else, you just have to expand the path to the destination folder on misfit after the colon.

1.3 Using rsync to create backups and to copy large data sets; use of tarballs

1. Download the tarball (a set of files bundled together in one compressed file) from http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/material/mM4_10048_pdfs/EXTREME_pdfs.tar.gz

2. Create a new directory on your local computer called 'astro_comp' in your home directory:
   `> mkdir astro_comp`

3. Now move `EXTREME_pdfs.tar.gz` into this new directory:
   `> mv EXTREME_pdfs.tar.gz astro_comp/`
   ...and change into `astro_comp/`

4. Decompress and un-tar the tarball, using:
   `> tar -zxvf EXTREME_pdfs.tar.gz`
   This should generate 71 files. Do you remember how to check the total number of files in `astro_comp/`?

5. Now lets make a complete backup of the directory `astro_comp/` and all the files in it by sending a copy to `misfit`. First change back into your home directory on your local machine. Then do:
   `> rsync -av --stats astro_comp/ misfit:astro_comp/`
   This generates a complete copy of your local directory `astro_comp/` in your `misfit`'s home directory with the same directory name there. Please have a look at the status summary output when `rsync` returns from doing its job. Actually, `rsync` can be used to backup data without having to transfer/copy all the files every time, but instead it only transfers changes. For instance, if you now run the same rsync command from above again, you will see that no files will be transferred (because nothing changed in your local copy). If, however, you were to modify one or more of the files in `astro_comp/` and you do the `rsync` again, only the modified files will be transferred. This is really useful also if you have many many big files to transfer to/from a computer or between different

hosts, because if some of the files can't be transferred within a day or so and/or the ssh connection closes for some reason, at least rsync will continue from the same point where it stopped to synchronise the folders and files, instead of starting the copy process from scratch. BTW: an important part of astronomical computing is to make regular backups of your files :)

1.4 Use of `nohup` and `nice`

1. Make a Bash script called `nohup_script.sh` that prints the current time every second for the next hour. Use a bash loop and `date` to print the current date and time. Let the code pause for 1 second within each loop increment, by using `sleep 1`.

2. Start the script to see if it works. It should print the date and time to the screen every second. Note that you can stop the script by pressing Ctrl-C.

3. Now start the script, but redirect the stdout and stderr to a file `shell.out`. While the script is running in one shell, open another shell and look into the file. You should see that every second a new line is appended with the current date and time to the end of the file.

4. Start the same script, but this time with `nohup`. Simply place `nohup` in front of the command that starts the script and `&` after the command. `nohup` is a wrapper for any command that you want to start such that it does not hang up (`nohup` means 'no hang up') when you log out. The final `&` at the end of a command line is used to start that process in the background, which means that you get the shell back when you hit enter, but the program keeps running in the background.

5. Use `tail -f shell.out` to show the end of the output file. It also follows any changes to the file (the `-f` option). You should see how the file grows and how every second, the date and time is appended as a new line to the file.

6. Now log out of the MSO server. Since we have started the script with `nohup`, it should not hang up after we logged out, but instead will keep running on the server even though we are logged out. So, lets login again and see if the script kept on writing the time to the file while we were away.

7. Caution: if you start something with `nohup`, it will keep running unless it's killed by the user or an admin. We made the script so it would stop after one hour, but lets simply kill the script by hand right now. To achieve that, we use the `ps` command, which shows all running processes on the host. Lets first add a customised version of `ps` to our `.alias` file to make it an easy task to show all our own processes (there is a whole bunch of other processes that are also running simultaneously on the server, but that we don't want to be listed). Add the following alias to your aliases:
   `alias myps='ps -elf |grep $USER'`
   and log out and log back in for this change to take effect.

8. Now do `> myps` in the shell. This should list your current active processes (see the pipe to `grep $USER`). Find the process ID that belongs to the running script that we started with `nohup` and kill the job using:
   `> kill -9 [PID]`
   where you have to replace '[PID]' with the process ID of the job to be killed. Alternatively, you can use

`> killall 'bash'`
which kills jobs based on their name. This will kill all your running bash scripts.

9. Note: You'd normally want to use `nohup` for jobs/scripts that take very long to run, for example over night or even longer and you don't want to keep an active shell open to the remote host while the script is running. This is when `nohup` is really useful. However, we should consider that other users might be running jobs or that a user could be physically sitting at that computer/host at the same time and trying to do some work. So a *nice* thing in this case is to use `nice`, which means that your job will only use CPU time (or compute power) when it is available. So in case the machine is under heavy load and running many processes at the same time, if you started your script with `nohup nice ./script.sh &`, then your job won't block the jobs of other users so much, because it waits for times when the machine is not under heavy work load.

2 Plotting data with gnuplot

2.1 Simple plots of data from ASCII text files

1. Lets make some simple plots first. Change into your `$HOME/astro_comp/` on the MSO server. Now look into the file `EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data` with `more` or `less` or `cat` (the file should still be there from the previous lecture or you can download it again). You will see that there is a header in the file with 10 lines (showing the mean, standard deviation and other statistical moments of the distribution function), followed by an empty line and then 4 columns with data in them.

2. Now go back to the shell and start gnuplot. First, plot column 1 against column 3 in the data file. Use this command:
`gnuplot> plot "EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data" using 1:3`
Note that you can also use auto-completion of the file name in gnuplot by hitting the tab key, just the same as in the shell. Ok, so this should show the density PDF in that file, which should look pretty close to a Gaussian.

3. Now replace "plot" with "p" and "using" with "u". This should do exactly the same as before, but is a bit more compact, i.e., no need to write out each gnuplot command, you can simply use the first letter in most cases and it will do the job.

4. Now plot the y-axis logarithmically. Do this with
`gnuplot> set log y`
and next enter the plot command from before again. Note that you can bring up the most recent last commands again, simply by using the up-arrow key; this should bring up the last few commands used. The same works in the Bash shell, which is very useful in case you made a mistake when typing the command, you can bring up the previous one instantly and correct only the bits of the command that didn't work or that you want to change—for example if you wanted to keep everything the same, but instead plot data from one of the other files in the directory.

5. Now plot on top of the previous plot, the respective data columns from file with number `*0060*`. Do this simply by adding to the end of the previous gnuplot command:
`, "EXTREME_hdf5_plt_cnt_0060_dens.pdf_ln_data" using 1:3`
This should now display two curves (or set of points) plotted on top of one another. You can add more lines/curves simply by appending more to the plot command. For example,

if we wanted to plot now also a constant line at $y = 10^{-3}$, we'd simply add '`, 1e-3`' to the end of the previous gnuplot command line and we should see the horizontal line at $y = 10^{-3}$.

6. You can change the x and y axis labels with:
   `gnuplot> set xlabel "log-density contrast"`
   `gnuplot> set ylabel "PDF"`
   ...then plot again and see if the axis labels have changed.

7. Ok, so this is all fine, but it usually doesn't look very nice. Gnuplot is really good for making a quick plot of some data, but making it pretty needs a bit more tweaking. Now lets see what can be done to make nicer figures, generate gnuplot scripts and automatically write postscript figures to a file.

2.2 Making scripts for gnuplot and generating postscript figures

1. Download the gnuplot script template `gnuplot.p` from http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/04/gnuplot.p (or via the link on the course web page).

2. Look into the file and go through each line step-by-step and see if you can make sense of the commands. Similar to Bash scripts, gnuplot just goes through the script line-by-line and executes the commands as if you entered them one-by-one in the interactive gnuplot mode. The advantage of the script is obvious: you don't have to write it all to the gnuplot prompt and you can very quickly regenerate the same plot and/or make little changes easily. So it's usually a good idea to script a plot that you make, in order to come back to it any time later, in order to reproduce the same plot. Running the script is then achieved simply by loading the script in gnuplot:
   `gnuplot> load "gnuplot.p"`
   which executes all the commands in the script line-by-line.

3. In this particular script, we have already switched to a different output type (or terminal); in this case to 'postscript' (`set term post eps`), which generates `.eps` figures. Such figure files may contain vector graphics (much preferred, because scalable), but can also contain pixel graphics (for example maps). Note that the graphics editor `inkscape` is very useful, because it can handle vector graphics and can be used to modify figures, keeping the advantages of vector graphics. In contrast, the popular `gimp` graphics editor can only handle pixel graphics. So while you can load an image containing vector graphics in `gimp`, it will be automatically converted to a pixel graphics image and thus loses all the advantages of scalable vector graphics.

2.3 3D plots with gnuplot

1. Lets make a simple 3D plot of the previous figure in gnuplot. Use:
   `splot "EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data" u 1:2:3`
   This shows the same density PDF as before, but now as a 3D plot. Note that you can turn the plot around by dragging it with your mouse.

2. Note that the 2nd column in the data file does not contain any useful data (it's all zeros), so it does not contain any information in the 2nd axis of the plot. Lets replace the 2nd column with the 4th column of the data file (which contains the cumulative distribution

function):
`splot "EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data" u 1:4:3`
This will now show a true 3D plot; i.e., the data in the 4th column of the file is now plotting along the 2nd axis of the 3D plot.

3. One can do more advanced things, for example changing the point types and adding a colour bar, e.g., by appending to the end of the previous line:
`splot ... u 1:4:3 with points palette pointsize 1 pointtype 6`

3 Making movies from a time series of plots/images

1. First we have to write a gnuplot script that writes out `.png` figures for each of the PDF data files (the time sequence of the PDF) in the tarball from Section 1.3 above.

2. We can use the basic gnuplot script from before, but in order to make png figures, lets change the gnuplot term to png:
`set term png size 800,480 enhanced font "Helvetica,14"`

3. Then after some other formatting definitions and setting (e.g., line types, log, key position, and axes labels; see script template from earlier) we make a gnuplot loop like this:
```
do for [i=20:90] {
    infile = sprintf('EXTREME_hdf5_plt_cnt_%04d_dens.pdf_ln_data',i)
    outfile = sprintf('frame_%04d.png',i-20)
    set output outfile
    p [-10:10] [1e-5:1] infile u 1:3 w lp ls 3 t sprintf('time=%04.0f',i)
    print outfile." created"
}
```

4. When you run the script, it should generate a list of figures `frame_0000.png`, `frame_0001.png`, ..., `frame_0070.png` and print to the screen that those files were created.

5. Now that we have the still frames, we can make a movie. The simplest `ffmpeg` command to make a movie from a bunch of still frames is:
`> ffmpeg -i frame_%04d.png movie.mpeg`
where you have files `frame_0000.png`, `frame_0001.png`, etc. previously generated with gnuplot.

6. You can play the movie file with `ffplay`. However, it won't play well over the network, so I recommend to copy it to your local computer (`scp`) and play it directly on your computer rather than in a window over the network.

7. So this is ok, but looks a bit pixelated. For a nicer movie, we have to increase the bit rate by adding the option `-b:v 5000k` to `ffmpeg`. This will greatly increase the bit rate and thus the quality of the movie output.

8. There are lots more advanced options in `ffmpeg`, for example cropping or extracting frames from movies and changing the encoder. It is one of the most powerful movie conversion/processing tools.

Use `man [command]` to bring up the manual page of a program/command to learn about possible command-line options for the program(s), or search the internet for answers/solutions.