



**Collaborative Computational Project for
Biomolecular Simulation**



Practical 2:

Running Biomolecular Simulations on the NGS
with the Portal



Outline

In this practical you will learn how to run biomolecular simulations on the National Grid Service using the AMBER simulation package using the NGS Portal. Multiple instances of a simulation of the Major Urinary Protein (MUP) will be run simultaneously in order to benchmark the AMBER code.

Starting structures, parameter and input files for these simulations have been prepared and are installed in your home directories on each training workstations. These files are for training purposes only, and CCPB offers no guarantee that these files are fit for research purposes. For details on how to set up protein simulations visit <http://www.ccpb.ac.uk/events/workshops>.

Stage 1: Authentication

In a web browser that contains your e-science certificate goto;

<https://portal.ngs.ac.uk>

If you already have credentials uploaded onto the MyProxy server and have authenticated yourself you may skip the to Stage 2 of this practical.

Step 1.1: Upload credentials onto the MyProxy server.

You may use two methods to upload your credentials onto the MyProxy server: the on-line Upload Tool or via the command line. It is simpler to use the Upload Tool, but if your institution employs a firewall that blocks the ports used by this tool you will have to use the command line method.

Both methods of upload require that you export your certificate from your browser, create userkey.pem and usercert.pem files from it, and store them in the .globus directory. Details of how to do this can be found at <http://www.grid-support.ac.uk/content/view/217/121/>.

Using the Upload Tool

In your browser click on [Authenticate](#) and then on [Launch New Upload Tool \(requires Java 1.6\)](#). The **MyProxy setup wizard** will open. To upload your credentials;

Click **Next**

Click **Next**

Click **Next**

Click **Test** to test your user key

Enter the PassPhrase you choose when you created the userkey.pem file

If the test is successful a tick will appear in the **Test button**

Click **Next**

Select **UK e-Science CA** from the **Pre-defined CAs:** box

Click <-

Click **Next**

Click **Next**

Click **Next**

Click **Next**

Click **Next**

Select **UK e-Science MyProxy Server** from the **Pre-defined MyProxys:** box

Click **Set Default....**

Click **Next**

Click **Upload**

Choose a username and enter it in the **Username** box (this is the username you will use to access the Portal)

Choose a passphrase and enter it in the **MyProxy PassPhrase** and **Re-enter PassPhrase** boxes (this is the PassPhrase you will use to access the Portal)

Enter the passphrase you choose when you created the userkey.pem file in the **Grid PassPhrase** box

Click OK

If your credentials were uploaded successfully close the wizard and skip to Step 1.2. If the upload failed then you will have to upload your credentials via the command line.

Upload via the command line

Connect to any NGS node using a GSI-SSH Terminal and copy the .globus directory from the local machine onto the NGS (details of how to do this are given in Practical 1).

Use the myproxy-init command to upload the proxy;

```
[ngsXXXX@ngs ~]$ myproxy-init -s myproxy.grid-support.ac.uk \  
-c ~/.globus/usercert.pem
```

You will be challenged to enter the passphrase you choose when you created the userkey.pem file, and asked to enter a MyProxy passphrase. The latter is a new passphrase that you will use to access the portal. Your username for accessing the Portal will be your NGS username on the machine you uploaded the credentials from, ngsXXXX.

Step 1.2: Log into the NGS Portal

In your web browser enter the following information into the Authenticate page;

MyProxy Server:	//myproxy.grid-support.ac.uk
MyProxy Username:	The username you choose when uploading your credentials
MyProxy Password:	The passphrase you choose when uploading your credentials

If you have successfully authenticated yourself, you should see a

You have Grid credentials (proxy) loaded

message towards the top of the page.

Stage 2: Uploading files onto the RAL node

In this practical we are going to run our simulations on the Leeds node, but demonstrate how the Portal can be used to stage jobs from other locations (including your home institution if you have a FTP or SFTP server).

To upload files onto the RAL node files via the Portal;

Click on [Browse_Host](#)

Select **gsiftp://ngs.rl.ac.uk:2811** from the **Connect To Resource...** drop down list

Click **Connect**

The **Active Connection** panel will now show a listing of all the files and directories in your home directory on the RAL node.

Click on [Upload](#)

Click **Browse**

In the file dialog box navigate into ~/ccpb/amber

Select the file **M01.md9**

Click **Open**

Click **Upload**

After the file has been uploaded a message stating "File uploaded ok: M01.md9" will be displayed. Repeat the process to upload the files **M01.top** and **md.in**.

Click **Back**

The three files that we have now uploaded are now shown in the directory listing.

Stage 3: Running Benchmarks on the Leeds node

Having now uploaded our input files onto the RAL node we are now going to perform a number of simulations simultaneously on the Leeds node to example how well the pmemd module of AMBER scales over a number of processors.

Step 3.1: Create Job Descriptions for each simulation

We are going to submit the same 10ps simulation of MUP to the Leeds node to run on 1, 2, 4, 8, 16 and 32 processors. Our first task is to create Job Descriptions for each of these 6 simulations. We will start with the 32 processor job;

Click [Applications](#)

Select **CCPb Workshop** from the **In Category:** drop down list

Click **Find NGS Applications**

Click [load](#) in the **AMBER (CCPb M01)** entry

The resulting page contains a template for an AMBER simulation that we are going to modify to perform the simulations;

In the **My Job Summary** pane click [Edit](#) on the **Job Name:** line

In the **Job Identification:** pane change **Name:** to "pmemd 32 processors"

In the **Job Resources:** pane select **//ngs.leeds.ac.uk:2119/pbs** from the **Submission Endpoint:** drop down list

In the **Job Type, Process Count, Wall Time:** pane change **ProcessCount:** to "32"

Click **Update Active Job Profile**

Click [Submit/Run](#) from the menu on the left hand side of the page

In the **My Job Summary** pane click [Edit](#) on the **Arguments:** line

In the text box modify the command that will be run to;

```
pmemd.MPI -O -i md.in -o M01_32.out -p M01.top -c M01.md9
```

We have modified the command to run the parallel version of pmemd (pmemd.MPI) installed on the Leeds node, and have specified that the output will be written to an output file whose name reflects that the job was run on 32 processors.

Click **Update**

Click [Submit/Run](#) from the menu on the left hand side of the page

Click [Save](#) from the menu bar at the top of the page

Click [Applications](#) from the menu on the left hand side of the page

Click **Find Personal Applications**

The job we have created is now shown in the list with a status of Unsubmitted.

Repeat the above process to create job descriptions for the 1, 2, 4, 8 and 16 processor jobs from the **AMBER (CCPb M01)** template, remembering to update the number of processors in the **Name:** and **ProcessCount:** records and the output file name in the **Arguments:** record. For the 1 processor job you will also want to change the **Arguments:** to use pmemd rather than pmemd.MPI.

Step 3.2: Stage the input files

Since all six of our simulations use the same input files we are going to stage the input files onto the Leeds node manually before submitting the jobs. If you are running the jobs that have their own input files then the Portal can stage them automatically when you submit the job. When submitting more than one job using the same input files you should stage the files manually so that there is no risk of the files being overwritten during the automatic staging process whilst they are being used by another job.

To stage the files;

- Click the [load](#) tag for any of run descriptions that you have just created
- Click [Edit](#) next to **Stage Data:**
- Check the tick boxes in column under the **X** symbol for all three entries
- Click on the **X** symbol and confirm removal of the files from the staging list
- Click [Browse For URI](#)
- Select **gsiftp://ngs.rl.ac.uk:2811** from the **Connect To Resource...** drop down list
- Click **Connect**
- Check the tick boxes shown for the files **M01.md9**, **M01.top**, and **md.in**
- Select **Add to Data Staging List** from the **Actions:** drop down list
- Click **Add to Data Staging List**
- Click **Stage in now** and confirm that you want to stage the files

Step 3.3: Submitting the jobs

Now that our job descriptions have been prepared and the input files have been staged we can start submitting our jobs for execution. To submit the first job;

- Click [Applications](#)
- Click **Find Personal Applications**
- Click [load](#) in the **pmemd 1 processor** entry
- Review the job description to check it looks okay
- Uncheck the **Stage all data when submitting job:** tick box
- Click **Submit My Job** and confirm that you want to run the job now

The job has now been submitted and the **Job Status:** entry now shows SUBMITTED. Clicking on **<Check Job Status** will update the job status and you should see PENDING if the job is queued or ACTIVE if the job is running.

Repeat the above process to submit the five other jobs, starting with the 2 processor job and finishing with the 32 processor job.

You may check the status of your jobs by clicking [load](#) for the job from the **Applications Repository**, and then clicking on **<Check Job Status**. When a job has finished its job status will be COMPLETED. The first job to finish should be the one running on 32 processors, after a few minutes, and the last should be the one running on 1 processor, which will take approximately 20 minutes.

Stage 4: Download results

Having performed the six simulations we now want to download the results onto our local machines to perform some analysis.

To download the results;

Click on [Browse_Host](#)

Select **gsiftp://ngs.leeds.ac.uk:2811** from the **Connect To Resource...** drop down list

Click **Connect**

The contents of the **Active Connection** directory panel at the bottom of the screen now show the files and directories stored in your home directory on the Leeds node, including the six output files that we have generated in this practical.

To download the six output files click on each file name in turn and confirm that you wish to download them.

Now that the files have been downloaded onto the local machine we will perform some analysis to determine the scalability of this simulation using the pmemd module of AMBER. In the following the results from simulations on 64 and 128 processors are also included.

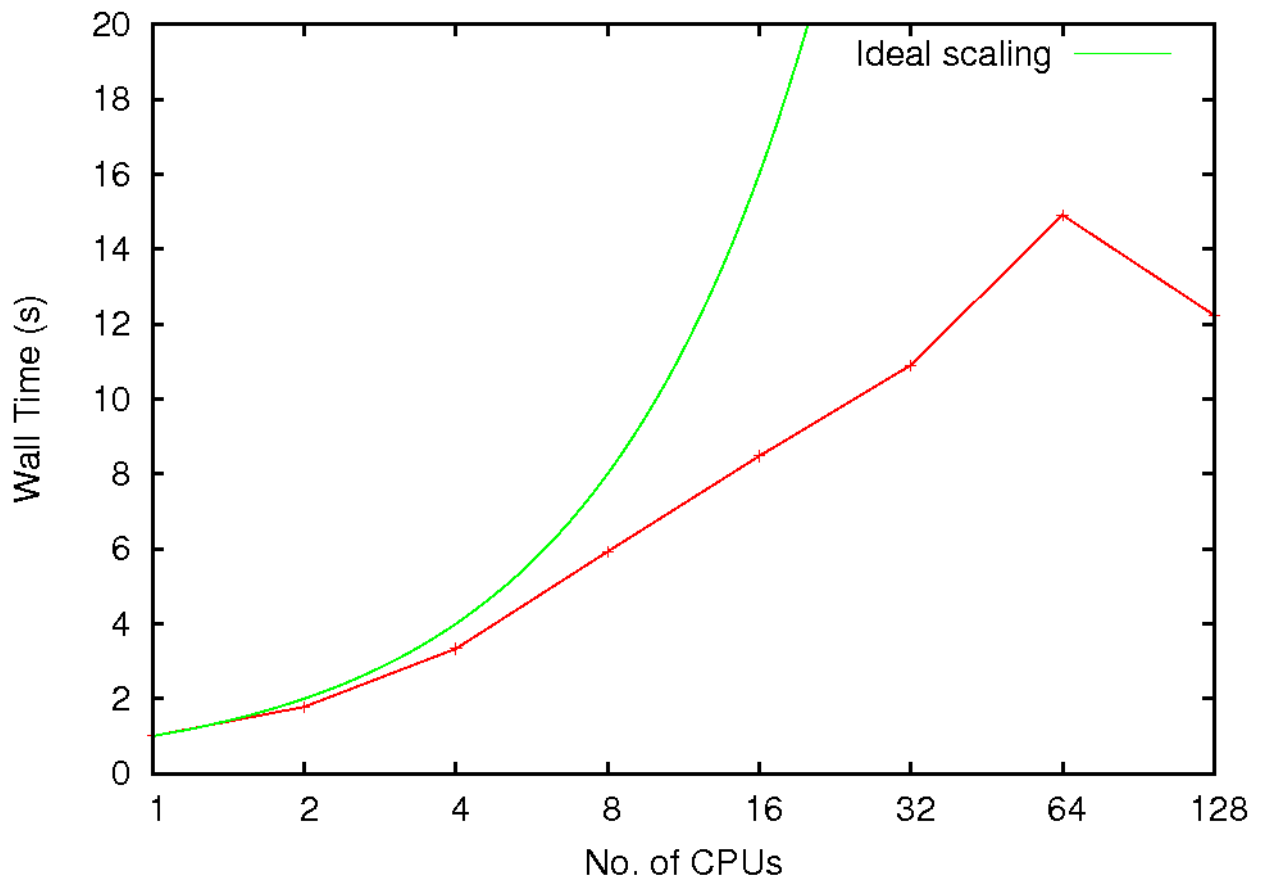
Open a terminal on the local machine

Execute the command;

```
$ grep "Total wall time" M01_*.out
M01_1.out: | Total wall time:      1297  seconds      0.36 hours
M01_2.out: | Master Total wall time:    729  seconds      0.20 hours
M01_4.out: | Master Total wall time:    389  seconds      0.11 hours
M01_8.out: | Master Total wall time:    219  seconds      0.06 hours
M01_16.out:| Master Total wall time:    153  seconds      0.04 hours
M01_32.out:| Master Total wall time:    119  seconds      0.03 hours
M01_64.out:| Master Total wall time:     87  seconds      0.02 hours
M01_128.out:| Master Total wall time:   106  seconds      0.03 hours
```

Here we are displaying the total wall time each simulation took. As is expected increasing the number of processors decreases the simulation time, from approximately 20 minutes for the job on 1 processor down to 1.5 minutes for the job on 64 processors. Interestingly as the number of processors is increased further the total time that the simulation takes increases, for 87 seconds on 64 processors to 106 seconds on 128 processors.

It is also of interest to plot these timing results to determine *Speed Up*, and compare this with *ideal scaling* behaviour, where ideal scaling means that a simulation running on twice as many processors should run in half the time;



This plot clearly shows that we are obtaining near ideal scaling when running on upto four processors (due to the hardware architecture of the compute nodes used). The observed scaling then begins to deviate significantly from ideal as the number of processors is increased further so that we only observe ~8 times speed up on 16 processors and ~15 times speed up on 64 processors.

To understand why the scaling behaviour deviates from ideal, and why the 128 processor job takes longer than the 64 processor job we will use the detailed timing output that AMBER provides to examine the amount of time spend performing the two most time intensive components of a parallel biomolecular simulation: calculating the non-bonded interactions, and transmitting data between the processors;

```
$ grep " Nonbond " *out
M01_1.out:|      Nonbond      1246.34   96.36
M01_2.out:|      Nonbond      639.57   88.05
M01_4.out:|      Nonbond      351.42   90.75
M01_8.out:|      Nonbond      187.78   87.07
M01_16.out:|     Nonbond      113.36   74.94
M01_32.out:|     Nonbond       62.63   53.17
M01_64.out:|     Nonbond       44.44   53.44
M01_128.out:|    Nonbond       26.62   26.39
```

The output from this command shows the wall time (in seconds) and the percentage of the total simulation wall time spent calculating non-bonded interactions. From this data we can clearly see that the wall time spent calculating non-bonded interactions decreases with increasing number of processors, as we would expect as there are more processors

working on the computationally challenging part of the simulation method. The percentage of the total wall time spent computing the non-bonded interactions also tends to decrease with increasing processors, indicating that the simulation is having to spend more time performing other tasks.

```
$ grep DataDistrib *out
M01_2.out:|      DataDistrib      60.72      8.36
M01_4.out:|      DataDistrib      17.37      4.48
M01_8.out:|      DataDistrib      18.62      8.63
M01_16.out:|     DataDistrib      32.65     21.59
M01_32.out:|     DataDistrib      52.21     44.33
M01_64.out:|     DataDistrib      36.43     43.81
M01_128.out:|    DataDistrib      72.51     71.88
```

Looking at the amount of time spent transferring data between processors shows that in general the amount of time occupied by inter-processor communication increases with increasing number of processors, such that when running on 128 processors it takes significantly longer for the communication tasks than the computationally demanding tasks.

It is this interplay of decreasing computational cost versus increasing communication costs that causes the deviation from ideal scaling behaviour, and for a sufficiently large number of processors will result in an increase in wall time as the increased communication cost completely negates the computational benefit of running on many processors.

From these results we can now determine the optimal number of processors to use when simulating MUP. Clearly running on 128 processors is wasteful as you will use in excess of twice the CPU time you would running on 64 processors. Whilst the scaling behaviour upto 64 processors is good, if you have a number of simulations to run it would be more efficient usage of the available CPU time to run four 16 processor, or two 32 processor jobs simultaneously. This is an important consideration when running on systems, like the NGS, where you have a limited amount of CPU time assigned to your account.

It should be noted that the scaling behaviour of biomolecular simulations is highly dependent upon the hardware being used, and upon the size of the system being simulated. A series of short benchmarking simulations should always be performed when running on a computational facility you have not benchmarked previously, and when simulating a system which is significantly different from any that you have benchmarked previously.